Evidence-Based Software Portfolio Management

Huijgens, Hennie

DOI
10.4233/uuid:f8fa946a-0178-40e7-bf9c-b91962698481

Publication date
2018

Document Version
Final published version

Citation (APA)
Huijgens, H. (2018). *Evidence-Based Software Portfolio Management*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:f8fa946a-0178-40e7-bf9c-b91962698481

Evidence-Based Software Portfolio Management

EVIDENCE-BASED SOFTWARE PORTFOLIO MANAGEMENT

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op 16 Februari 2018 om 12:30 uur
door

Hennie HUIJGENS

Master of Science in Information Management - Universiteit van
Amsterdam
geboren te Alphen aan den Rijn, Nederland

Dit proefschrift is goedgekeurd door de promotors:
Prof. dr. A. van Deursen
Prof. dr. ir. D.M. van Solingen

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof. dr. Arie van Deursen	Technische Universiteit Delft, promotor
Prof. dr. ir. Rini van Solingen	Technische Universiteit Delft, promotor

Onafhankelijke leden:

Prof. dr. Egon Berghout	Rijksuniversiteit Groningen
Prof. dr. ir. Marijn Janssen	Technische Universiteit Delft
Prof. dr. Magne Jørgensen	University of Oslo, Simula, Norway
Prof. dr. Emerson Murphy-Hill	North Carolina State University, USA
Prof. dr. Claes Wohlin	Blekinge Institute of Technology, Sweden

The work in this thesis has been carried out at the Delft University of Technology, supported by Goverdson.

All photos published in this thesis are taken from Unsplash, licensed under Creative Commons Zero. Cover photo by Derek Thomson on Unsplash.

IBSN 978-94-028-0932-9

Author email: hennie@goverdson.nl

ROSALIND
A traveler. By my faith, you have great reason to be sad. I fear you have sold your own lands to see other men's. Then to have seen much and to have nothing is to have rich eyes and poor hands.

JAQUES
Yes, I have gained my experience.

ROSALIND
And your experience makes you sad. I had rather have a fool to make me merry than experience to make me sad— and to travel for it, too.

As You Like It (1599), Shakespeare

I believe if there's any kind of God it wouldn't be in any of us, not you or me but just this little space in between. If there's any kind of magic in this world it must be in the attempt of understanding someone sharing something. I know, it's almost impossible to succeed but who cares really? The answer must be in the attempt.

Julie Delpy, in Before Sunrise (1995)
by Richard Linklater

Photo by Ryan Lum

Acknowledgments

Two quotes kick off this thesis. In the first one Rosalind – one of Shakespeare's most beloved heroines, an intelligent woman with a strong will – complains to the hopeless melancholy, on the sidelines operating Jaques how terrible it must be to have a lot of experience. Rosalind explains Jaques that experience should make him sad, and that she prefers a fool to make her merry.

How recognizable in my daily job as a software analytics specialist, where one of my main activities consists of the collection and analysis of experiences on software development activities, presented as evidence that supports future decision-making. Am I Jaques, a professional outsider who delights in being sad? Or do I, alike Rosalind, prefer a fool (or a tool) that makes me happy?

Regardless of the choice Hans Jägers, professor emeritus at the University of Amsterdam, plays the role of Duke Senior, the rightful ruler of the dukedom in which the play is set. In his role of Chairman of the Board of Examiners of the Executive Master in Information Management from the University of Amsterdam, that I finalized in 2010, he raised the question ‘Why are you not going to do a PhD on this?’ I gave it some time to digest, and talked things over with Isabel, my wife, after which I was left with the question ‘Yes, why not?’

It took me some time to find a professor who dared to join me in the play, but Rini van Solingen was the best opponent I could wish for, and soon Arie van Deursen also joined, fascinated by the experience in the form of a rich set of industry data that I had displayed as attributes on the stage. Rini and Arie, thanks so much for your inspiring support, your professional guidance, and the trust you have given me to find my way in a play without a boarded script. Many thanks! You were the best companions I could think off.

What I probably liked most during my PhD was the collaboration on paper writing with some of the best fellow researchers in the field. In a way, they helped me preparing this thesis after all. Thanks a lot, to you all.

Working within the inspiring and international environment of the Software Engineering Research Group at the Delft University of Technology was an unforgettable experience for me. Thank you all very much for the conversations, the inspiring lunches and the occasional cocktail parties.

I realize that as a ‘practitioner at age’ I sometimes was an odd man out in the research community, but I am well aware that precisely this mix of seniority, practice and research, was a huge breeding ground for research which I often could directly apply in an industrial context. I never could have performed research in the way I did, without the ongoing support of the companies that hired me as a consultant, and the colleagues and executives that joined in the collection and analysis of software project data. Thank you so much for your trust and generosity.

The second quote that I choose to illustrate the process of my PhD is more recent. It is about the importance of the space between things. That’s where the magic in this world is to be found, in the attempt of understanding someone sharing something. The answer must be in the attempt.

‘Just try’. That is what I learned from my darlings at home. Isabel told me that she really enjoyed seeing me doing my PhD in such a cheerful, bright and autonomous way. ‘What I liked most, was that you gave me the idea that doing a PhD is a piece of cake,’ she said to me when I was almost done, and my son Julian and my daughter Bloem could only agree with her.

I cannot think of a better way to look back on an inspiring and enjoyable four years.

Thank you all for your support and love.

Hennie Huijgens
Amsterdam,
September, 2017

Table of Contents

Based on the large amounts spent by software companies to develop new and existing software systems, we argue that an evidence-based approach that focuses on a software portfolio as a whole should be in place to support decision-making.

Photo by Martin Ceralde

1. Introduction

This thesis addresses software projects; more specifically, it is about comparing software projects among themselves. How can companies learn from their good and bad projects as input for future software engineering activities? What software projects can be seen as good practice, and thus as an example for others? And what projects can be looked upon as bad practice, and how should companies improve these? Yet, such a comparison is not straightforward; when looking at the different software projects that are undertaken in software companies, none are equal.

An unambiguous definition of the concept of a 'project' is difficult to find. Nokes (2007) for example, considers a project as 'a temporary endeavor designed to produce a unique product, service or result with a defined beginning and end (usually time-constrained, and often constrained by funding or deliverable) undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value'.

Although more or less formal projects, as stated above with a defined beginning and end, are still to be found in many software companies, software engineering practice did change rapidly since the start of the millennium (Fitzgerald & Stol, 2015) (Boehm, 2006a) (Dingsøyr & Lassenius, 2016).

Software engineering nowadays is an ongoing process of development and maintenance of software solutions, covering the lifecycle of a software system. Because of this holistic, lifecycle-driven way of observing, projects can be looked upon as a huge variety of approaches, varying from traditional waterfall projects to iterative releases by DevOps-teams (Fitzgerald & Stol, 2015). Software projects in such modern environments are often less formalized, and might better be defined as software deliveries.

In order to accommodate the range from traditional, plan driven projects to iterative, agile deliveries in our approach, we use Nokes' definition (2007), with the addition that software projects are performed in a variety of different ways, ranging from traditional plan-driven to iterative deliveries. Therefore, within this thesis, we use the concepts of project and delivery as equivalent expressions.

It is important for software companies to know which of their projects are successful and which are not, and what are the backgrounds of success and failure. Software is eating the world (Andreesen, 2011), and information technology is the largest production factor for many organizations (Verhoef, 2002). The software projects of the companies that we studied within the scope of this thesis represent on a yearly basis between 10 and 300 million Euros, and even larger information technology budgets are found in related work (Verhoef, 2002). At the same time managing such huge spending on software projects is challenging due to the fact that software has become so complex and it evolves so quickly that we fail to keep it under control (Huisman et al., 2016). Software companies spend a significant amount - some up to seventy-five percent - of their IT budgets maintaining legacy systems (Arnold & Braithwaite, 2015) (Gangadharan, Kuiper, Janssen, & Luttighuis, 2013).

Knowing the characteristics of projects, especially in large and hybrid portfolios, that usually in addition to newly built applications also include old and complex legacy systems, is important to isolate good from bad practice and to identify aspects for improvement. Good and bad practice can only be distinguished by looking at the whole of a company's software projects. That is why we argue that management at a *portfolio* level is important, where a focus on cost, time, and quality seems obvious (Boehm, 1984) (Kan, 1995).

Where project management focusses mainly on *doing projects right*, project portfolio management is focused on *doing the right projects* (Reyck, et al., 2005). Analogies that build on financial-portfolio theory are not new (McFarlan, 1981) (Dye & Pennypacker, 1999). More recent examples of this analogy with a portfolio of financial assets, trying to improve the performance of the portfolio by balancing risk and return are (Jeffery & Leliveld, 2004) and (Verhoef, 2002). The latter argues that 'the heart of security portfolio management is to monitor, control, and optimize the security selection process', where he defines quantitative IT Portfolio management as 'considering the quantitative aspects of IT development, operations, maintenance, enhancements, and renovation for bespoke software systems' (Verhoef, 2002). (Reyck et al., 2005) consider project portfolio management as 'the entire portfolio of projects a company is engaged in, in order to make decisions in terms of which projects are to be given priority, and which projects are to be added to or removed from the portfolio'.

Often software portfolio management is defined as considering aspects of information technology development, operations, maintenance, enhancement, and renovation for bespoke software systems, where the management scope is limited to quantitative aspects, such as cost, time, and quality of estimations (Boehm, 1984) (Kan, 1995).

Yet, are projects that cost relatively little and are quickly delivered more valuable than more expensive ones that took relatively long? And are stakeholders more satisfied when cost is low and time is short? Or are these relations more complicated?

To examine these questions, we follow in the context of this thesis the existing definition of Verhoef (2002), but we extend these with *qualitative* aspects, in addition to *quantitative* ones. We define *software portfolio management* as 'to monitor, control, and optimize the quantitative aspects of IT development, operations, maintenance, enhancements, and renovation for bespoke software systems, in relation to the qualitative aspects stakeholder satisfaction and delivered value'. However, several problems occur with contemporary software portfolio management, starting from cost, time, and quality related issues to the determination of value and stakeholder satisfaction.

1.1. Problems within software portfolio management

1.1.1. Success and failure defined exclusively at a project level

Problem 1: Success and failure are usually defined related to the estimated budget and time of a project. They are not simple and unambiguous to define at a portfolio level.

Often software companies define success or failure of their software projects related to they were not delivered on time, within cost, and with all specified functionality (International Standish Group, 1994). Supported by many critical reviews of such an approach (Jørgensen & Moløkken-Østvold, 2006) (Glass, 2006) (Eveleens & Verhoef, 2010), in this thesis we look at project success and failure from a portfolio point of view.

Based on strong correlations occurring between cost, duration, quality, and size of software projects (Boehm, 1984) (El Emam & Günes Koru, 2008) (Boehm, Abts & Chulani, 2000a) (Heemstra & Kusters, 1991) (Bhardwaj & Rana, 2016), we examine whether the meaning of success and failure is to be found in software portfolios as a whole, instead of in individual projects.

1.1.2. Linking legacy evolution with new functionality

Problem 2: Linking evolution of legacy systems with development of new functionality is a challenge within portfolio management

Software portfolio management is about monitoring, controlling, and optimizing the software engineering process within a specific software portfolio, where such a portfolio exists from a variety of software related activities such as adding, changing, and deleting software functionality, and maintaining existing applications. Managing such existing applications, and especially linking the development of new software with evolution of legacy systems is a big challenge for many companies (Boehm, 2006b) (Deursen, Klint & Verhoef, 1999). The effects of legacy and maintenance, including accompanying delivery approaches, are not clear when looked upon from a portfolio point of view.

1.1.3. Pricing and estimation relies heavily on expert opinions

Problem 3: Effort and cost estimation, and pricing of software deliveries rely heavily on expert opinions and often are only to a limited extent transparent and evidence-based.

While many studies can be found on estimating the cost (or effort) of software deliveries (Jørgensen & Shepperd, 2007), only a handful exists about the price thereof. For pricing purposes in a commercial context where software delivery is partly performed by suppliers, most companies rely heavily on expert judgment (Boehm, 1984) (Jørgensen, 2004). This is not always a successful approach (Moløkken & Jørgensen, 2003), and software development is often characterized by high cost and schedule overruns (Verhoef, 2002) (Glass, 2002), despite a huge number of tools developed over time to support project estimation (Boehm, 1984) (Abran, Silva, & Primera, 2002b) (Gencel & Demirors, 2008) (IFPUG, 2009).

Our observation in industry (at least in the companies that are subject of this thesis) is that a purely statistical method – where pricing is solely based on data analysis and not at expert opinions - is not used. Nevertheless, from a portfolio point of view, statistics seem a natural tool to predict prices of future software deliveries based on historic data of finalized deliveries in a specific software company.

1.1.4. Cost and effort are used as equivalent

Problem 4: Effort and cost of software deliveries are often seen as equivalent. The relation between both metrics seems complex and is difficult to understand.

Two additional problems with determining the concept of good and bad performance of software projects are the non-availability of historic project data in many software companies, and the confusing fact that cost and effort are often looked upon as equivalent, e.g. (Radliński, 2011) (Jeffery, Ruhe, & Wieczore, 2000) (Pendharkar & Rodger, 2009) (Czarnacka-Chrobot, 2009). At the best, effort is assumed to be a good proxy for cost, where the emphasis seems to be more on effort, and less on cost. Regarding the non-availability of historic data it is striking that many generic benchmarks are available for software projects (Jones, 2011) (Menzies & Zimmermann, 2013), but that cost data is missing in most of them. And that seems strange. A software company's project portfolio is built from differently organized cost structures, and many decision makers use cost as a major indicator for decisions.

1.1.5. Value and Stakeholder Satisfaction only limited in scope

Problem 5: The relation between stakeholder satisfaction and perceived value on the one hand and software project performance in terms of time, cost, and quality on the other is not clear and hinders efficient steering on value optimization within software portfolio management.

Traditionally software portfolio management is about quantitative aspects such as cost, time, and quality (Verhoef, 2002) (Boehm, 1984) (Kan, 1995). How these relate to the backgrounds of success and failure of software deliveries, especially regarding stakeholder satisfaction and value, remains often unclear. Although quite some research has been performed on aspects of value and software projects (Boehm, 2003) (Biffl, Aurum, Boehm, Erdogmus,

& Grünbacher, 2006) (Faulk, Harmon, & Raffo, 2000)(Dingsøyr & Lassenius, 2016) (Agarwal & Rathod, 2006) (Bryde, 2005), most of these approaches seem poorly adopted in industrial software project management settings, although agile development approaches have a positive impact on the focus on value as an important metric (Dingsøyr & Lassenius, 2016). Jørgensen (2016) mentions that a focus on client benefits as a success criterion is particularly important, because only weak correlations are found on other dimensions, such as *being on time* and *being on budget*.

1.2. Research Goal and Research Questions

Based on the problems as described in the former Subsection, we define as the goal for our research to help software companies understand how software portfolios perform in terms of time, cost, quality, value creation, and stakeholder satisfaction, to maximize the benefits of software deliveries. For that purpose, we developed the following five research questions:

1.2.1. Success and Failure Factors for Software Projects

RQ1: What success factors and failure factors affect software project portfolio performance?

The first research question is related to a set of core metrics of software deliveries (Kan, 1995), respectively size, cost, duration, and number of defects, and elaborates on how they interrelate with each other. These four core metrics are used to build a model for comparison of software deliveries of all sorts within one or more software project portfolio's, and sets the basis for the benchmarking approach that we use in the remaining research. The goal of the first research question (RQ1) is to identify success factors and failure factors that affect the performance of software project portfolios.

1.2.2. New Developments, Maintenance, and Legacy

RQ2: What actions can be taken to increase project performance when running a software project portfolio with development of new functionality and maintenance of legacy systems involved?

The second research question (RQ2) relates to the application of the findings of RQ1 when steering on improvement of the performance of a hybrid software project portfolio, containing a mix of new development projects and maintenance and enhancements on existing legacy systems. However, in this research question we elaborate further on the specific effects related to evolution of legacy systems in a portfolio, and the relation between building new functionality and enhancement and maintenance on existing (legacy) systems.

1.2.3. Evidence-Based Pricing of Project Proposals

RQ3: How can an empirical, evidence-based pricing approach for software engineering, be used as a single instrument (without expert judgment), to create cost transparency and cost and time improvements?

The third research question (RQ3) is set up to examine how to use a pricing approach for project proposals in a distributed outsourcing context, based on an empirical, evidence-based way to determine fixed prices of software projects. Our main goals were to investigate whether such an approach helps to improve transparency and stakeholder satisfaction, and cost and duration improvements.

1.2.4. Cost and Effort in Measurement Repositories

RQ4: How do data repositories compare on size, cost, effort, duration and number of defects, and how can differences be explained?

Many software companies use benchmark repositories to support estimation and pricing of their software projects. Research question number four (RQ4) examines differences regarding cost and effort of software deliveries between our EBSPM repository and the ISBSG repository, a commonly used source for effort and cost prediction and benchmarking in industry.

1.2.5. Stakeholder Satisfaction and Perceived Value

RQ5: How do stakeholder satisfaction and perceived value relate to software project performance?

Finally, the fifth research question (RQ5) involves *stakeholder satisfaction* and *perceived value* in the equation, and examines overall correlations

Table 1.1: Mapping of Research Questions on the Chapters in this thesis.

Research Question	Chapter: 3	4	5	6	7
RQ1: What success factors and failure factors affect software project performance?	√	√			
RQ2: What actions can be taken to increase project performance when running a software project portfolio with new developments and maintenance of legacy systems involved?	√	√			
RQ3: How can an empirical, evidence-based pricing approach for software engineering, be used as a single instrument (without expert judgment), to create cost transparency and cost and time improvements?			√		
RQ4: How do data repositories compare on size, cost, effort, duration and number of defects, and how can differences be explained?	√	√		√	
RQ5: How do stakeholder satisfaction and perceived value relate to software project performance as identified in RQ4?	√				√

between all metrics involved in the EBSPM approach. The main goal behind this research question is to examine whether *good practice* and *bad practice*, as defined by RQ1, fits with the way stakeholders of projects (e.g. developers, decision-makers, business executives, customers) experience project- and portfolio performance.

Table 1.1 gives an overview of the research questions and how these questions map on the chapters in this thesis.

1.3. Research Method and Evaluation

Driven by our ambition to provide as much value as possible with our research to software-intensive companies, we opt for a research method that proposes a model that subsequently is evaluated through empirical studies, such as case studies, and surveys (Wohlin et al., 2000). In our model, we combined two aspects that largely determine our research setup, (1) an evidence-based approach, and (2) a focus at a software company's portfolio of software deliveries.

1.3.1. An evidence-based approach with EBSE as an example

For the approach that is presented in this thesis gratitude is owed to (Kitchenham, Dybå, & Jørgensen, 2004), who presented a method to support the structured and evidence-based decision-making in the field of software engineering. Their *Evidence-Based Software Engineering* (EBSE) approach was an inspiration on how to conceive a practical and experience-based method for software companies to help them to better monitor and control their software project portfolios.

In turn they derived their approach from *Evidence-Based Medicine*, a similar method which was developed in the medical field, where medical researchers found that failure to organize existing medical research did cost lives, and that clinical judgement of experts compared unfavorably with the results of systematic reviews (Kitchenham et al., 2004). Thinking from the research question whether an evidence-based paradigm is feasible for Software Engineering too, they started an analogy-based comparison in order to provide the means by which present-day best evidence from research can be integrated with best practices from industry and human values in the decision-making process regarding the development and maintenance of software (Kitchenham et al., 2004).

1.3.2. A holistic view at a company's software portfolio

Based on the large amounts spent by software companies each year to develop new and maintain existing software systems, we argue that an evidence-based approach should be in place to support decision-making on their software activities. Many studies are to be found that guide decision makers on aspects of time, cost, and quality of software projects, among others on software estimation and benchmarking of software engineering activities. However, we recognize two important developments in contemporary software development, that require, in addition to EBSE, a new and complementary approach aimed at the decision-making on software projects.

Decision-making should be looked upon from a holistic perspective. Analysis of the finalized software projects in our research repository shows that the included software companies spent each year between 10 and 300 million Euros on software engineering, largely depending on their company size. We observe large variations between software projects, in size, in time, in the number of defects, and especially in cost. Some projects perform outstanding, and might be looked upon as *good practice*, yet others perform much worse, and might be characterized as *bad practice*.

When evaluation focuses at the level of single projects, software companies cannot tell whether it performed better than average or worse than average. Learning from experience is difficult in that case. Based on the huge differences in project performances that we found, we argue that the economic aspects of software engineering should best be looked upon from a holistic – thus a portfolio – point of view, instead from an individual – thus a single project - one. In other words, when steering on project performance *all* projects in scope of a company's portfolio should be considered.

When decision makers use results of studies performed on data from other companies, such as algorithms and prediction models, a major risk is in place that company specific effects on the project performance might be excluded. To mitigate this risk, we argue that software companies should best collect their own historic project data, as a valuable source to support decision-making on future software activities.

1.3.3. EBSPM as the proposed model

With these two principles in mind – evidence-based decision-making *and* a focus on a company's own software portfolio as a whole – we developed our model, and named it *Evidence-Based Software Portfolio Management* or EBSPM. EBSPM can be described as a model aimed at supporting decision makers of software projects, based on analysis of the entire software project portfolio of in the past completed projects in their own organization. EBSPM builds on two starting points.

Firstly, software companies should always collect their own historic data, instead of relying fully on cross-company datasets for estimation and benchmark purposes (Jeffery, Ruhe, & Wieczore, 2001) (Briand, Langley, & Wieczorek, 2000) (Wieczorek & Ruhe, 2002) (Lokan & Mendes, 2006) (Minku, Mendes, & Ferrucci, 2015) (Mendes, Lokan, Harrison, & Triggs, 2005) (Garre, Cuadrado, Sicilia, Charro, & Rodríguez, 2005) (Minku, 2016). However, for companies that have limited projects, it might be a second best option to compare their own historic data with external data sources, keeping in mind that large differences might occur.

Secondly, to understand the concepts of good and bad performance it is important to study software portfolios as a whole. Besides projects that score well regarding time, cost, and quality, also projects occur that score poorly on those aspects. Knowing and understanding a good balance between *all* deliveries in scope of a software portfolio is important when taking decisions.

To support these starting points, EBSPM is built around three key components. The first is an approach to collect, analyze, and benchmark finalized software deliveries, based on time, cost, quality, value-creation, and stakeholder satisfaction. The second is a research repository holding historic data from approximately 500 finalized software deliveries in different companies and business domains. The final component is a performance dashboard that visualizes successful software deliveries (also called *good practice*) and less successful deliveries (also called *bad practice*) within a software portfolio.

1.3.4. Evaluation of EBSPM through case studies and surveys

The majority of our research has been applied in close cooperation with software companies in industrial practice. To maximize the practical application of our research in an industry context, we opt for an iterative approach with short feedback loops. Such an approach makes it possible to align our research goals tightly with the strategic and tactic goals of the companies that participate and to change plans whenever this is needed for practical reasons. We designed a series of small, practically oriented steps, which lead to results which were directly useable in a practical context, and which could be incorporated to self-contained scientific publications. Each study was performed according to an empirical strategy that fitted best to its practical context, to ensure maximum alignment with the specific industry environment.

Table 1.2 gives an overview of the empirical strategies (Wohlin et al., 2000) that were applied in the different studies, where each chapter represents one specific study. As can be seen the majority of studies are performed as case study or a survey, or in many cases a combination of both. Two case studies were single case studies and one was performed as a multiple case study, performed in two different companies. All case studies combined both quantitative and qualitative methods (Yin, 2008) (Runeson, Host, Rainer, & Regnell, 2012).

As an instrument for qualitative research we made in three studies use of electronic surveys to question stakeholders of finalized software projects

Table 1.2: Mapping of empirical strategies on the Chapters in this thesis.

Empirical strategy	Chapter:	2	3	4	5	6	7	8
Survey				Q/I	Q		Q	
Case Study				S	S		M	
Experiment or replication								
Data Analysis Study			D	D	D	D	D	

S = Single Case Study, M = Multiple Case Study, Q = Survey with Electronic Questionnaires, I = Survey with Interviews, D = Data Analysis Study (Quantitative)

about their experiences. In one study, this qualitative approach was supplemented with structured interviews with stakeholders.

Although all studies include a data analysis component, two studies, represented in Chapters 3 and 6, do not include a qualitative study, and can be characterized as typical data analysis studies. In Chapter 3 we analyze an existing repository that forms the basis for our EBSPM research repository for causes behind success and failure of software projects (Huijgens, van Solingen, & van Deursen, 2014c). In Chapter 6 we compare our EBSPM research repository with a subset of projects from the ISBSG repository on the relations between effort and cost of software projects (Huijgens, van Deursen, Minku, & Lokan, 2017c).

Within the scope of this thesis no experiment, quasi-experiment, replication, or structured literature review was performed.

1.4. Thesis Outline

Each chapter in this thesis represents a study that was performed in close cooperation with one of the four different software companies that participated in our research. The first two are both large Dutch banks – identified in this thesis as *Bank-A* and *Bank-B* – with complex software project portfolios, operating in the midst of an enterprise-wide transformation from a plan-driven (waterfall) development approach towards an agile (Scrum) way of working. The third is a midsized Belgian telecom company – identified in this thesis as *BelTel* – moving from a plan-driven approach towards Scrum-teams, together with a strategic transformation from a European main supplier towards *IndSup-A*, an Indian supplier of development teams. The fourth is a relatively small Dutch Billing software company that runs globally distributed development teams in close cooperation with *IndSup-B*, a Netherland's-based supplier that runs development teams in India, combined with an agile (Scrum) way of working.

1.4.1. Origin of Chapters

In every individual chapter, a specific subject of EBSPM is addressed. Each chapter in this thesis, except for the introduction of EBSPM in Chapter 2, and the conclusions and future research in Chapter 8, is based on a peer-reviewed publication at a conference or in a journal, and can thus be read in separation. The author of this thesis is the first author of all publications, however, all papers – except for a paper for a doctoral symposium and a tool description that are combined into Chapter 2 – are written in close cooperation with others.

Chapters 3 to 7 are all directly based on the published papers. We adjusted the layout of the original papers to include them as chapters in this thesis. The *cost duration matrix*, a central, and major part of the EBSPM-tool is described in detail in Chapter 2. In order to improve the readability of this thesis, we shortened the descriptions of the matrix in the Chapters 3, 4, and 7, and included a reference to the general description in Chapter 2. Furthermore, we adjusted the numbering of research questions to match them with the chapter numbering; in case of only one research question in a chapter, we did not number it.

Chapter 2: A birds-eye view on EBSPM

In Chapter 2 we briefly introduce EBSPM as a model to support experience-driven portfolio management in software companies. The text in this chapter is based on two publications: 1) *Evidence-Based Software Portfolio Management* in the proceedings of the 2014 doctoral symposium of the 9th International Symposium on Empirical Software Engineering and Measurement (ESEM 2015) (Huijgens, 2015a), and 2) *Evidence-based software portfolio management: a tool description and evaluation* in the 2016 proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE 2016) (Huijgens, 2016a).

Unlike all following chapters, we adapted the text of the original publications where applicable, to make Chapter 2 suitable as an introduction for readers who want to get a brief overview of the EBSPM-model and the accompanying EBSPM-tool.

Chapter 3: About Good Practice and Bad Practice

Chapter 3 appeared as *How to build a good practice software project portfolio?* in the 2014 companion proceedings of the 36th International Conference on Software Engineering (ICSE SEIP 2014) (Huijgens, van Solingen, & van Deursen, 2014c). In this chapter, we describe how analysis of the initial data set of 352 finalized software projects, led to an inventory of seven success factors and nine failure factors for software deliveries.

Chapter 4: EBSPM in a legacy context

Chapter 4 highlights the application of EBSPM in a legacy context in industry. We performed a mixed, retrospective case study with in-depth interviews with stakeholders on a series of nine software releases and eight single once-only releases, all performing on a single, legacy software system, in a West-European telecom company. This chapter was published as *Success factors in managing legacy system evolution: a case study* in the proceedings of the 38th International Conference on Software and Systems Process (ICSSP 2016) (Huijgens, van Deursen, & van Solingen, 2016d).

Chapter 5: EBSPM as a basis for Project Pricing

In Chapter 5 we highlight a case study in a Belgian telecom company where the results of linear regression models based on data from completed software deliveries, were used for the preparation of fixed price project proposals in a strategic, long term outsourcing context with an Indian supplier. This chapter was published as *Pricing via functional size: a case study of 77 outsourced projects* in the proceedings of the 9th International Symposium on Empirical Software Engineering and Measurement (ESEM 2015) (Huijgens, Gousios, & van Deursen, 2015c).

Chapter 6: A comparison of two Software Project Repositories

In Chapter 6 we examined the characteristics of the software delivery data that we collected over time in the EBSPM repository. We compare the EBSPM repository with a commonly used repository that is maintained by the International Software Benchmarking Standards Group (ISBSG, 2014). This chapter is published as *Effort and Cost of Software Engineering: A Comparison of Two Industrial Data Sets* in the proceedings of the 21st International

Conference on Evaluation and Assessment of Software Engineering (EASE 2017) (Huijgens, van Deursen, Minku, & Lokan, 2017c).

Chapter 7: Stakeholder Satisfaction and Perceived Value

Chapter 7 highlights the addition of two important metrics for agile software delivery to the EBSPM framework, *stakeholder satisfaction* and *perceived value*. In an extended case study in two different software companies we examine correlations between the metrics that we collected over time in our EBSPM research repository. This chapter is published as *The Effects of Perceived Value and Stakeholder Satisfaction on Software Project Impact* in the journal Information and Software Technology (Huijgens, van Deursen, & van Solingen, 2017d). This journal paper is an extended version of a best-paper award winning publication at the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE) (Huijgens, van Deursen, & van Solingen, 2016c).

Chapter 8: Conclusions

Finally, in Chapter 8 we inventory the contributions of our research. We discuss findings and threats to validity. We outline the implications for research, the software engineering industry, and education, and finally we draw conclusions based on the collections of chapters in the thesis.

1.4.2. Additional tests summarized in an addendum

Due to the basic principle that each chapter in this thesis - except for the introduction of EBSPM in Chapter 2, and the conclusions and future research in Chapter 8 - is based on a peer-reviewed publication at a conference or in a journal, the thesis itself reflects the four-year history of its development. This can especially be seen in the application of statistics in this thesis. Two main aspects play a role here. Firstly, the level of knowledge of the author of this thesis on software analytics, and more specifically the application of statistics matured during this period. Secondly, the knowledge of software analytics and the use of statistics within the discipline of software engineering research itself developed to a higher level too.

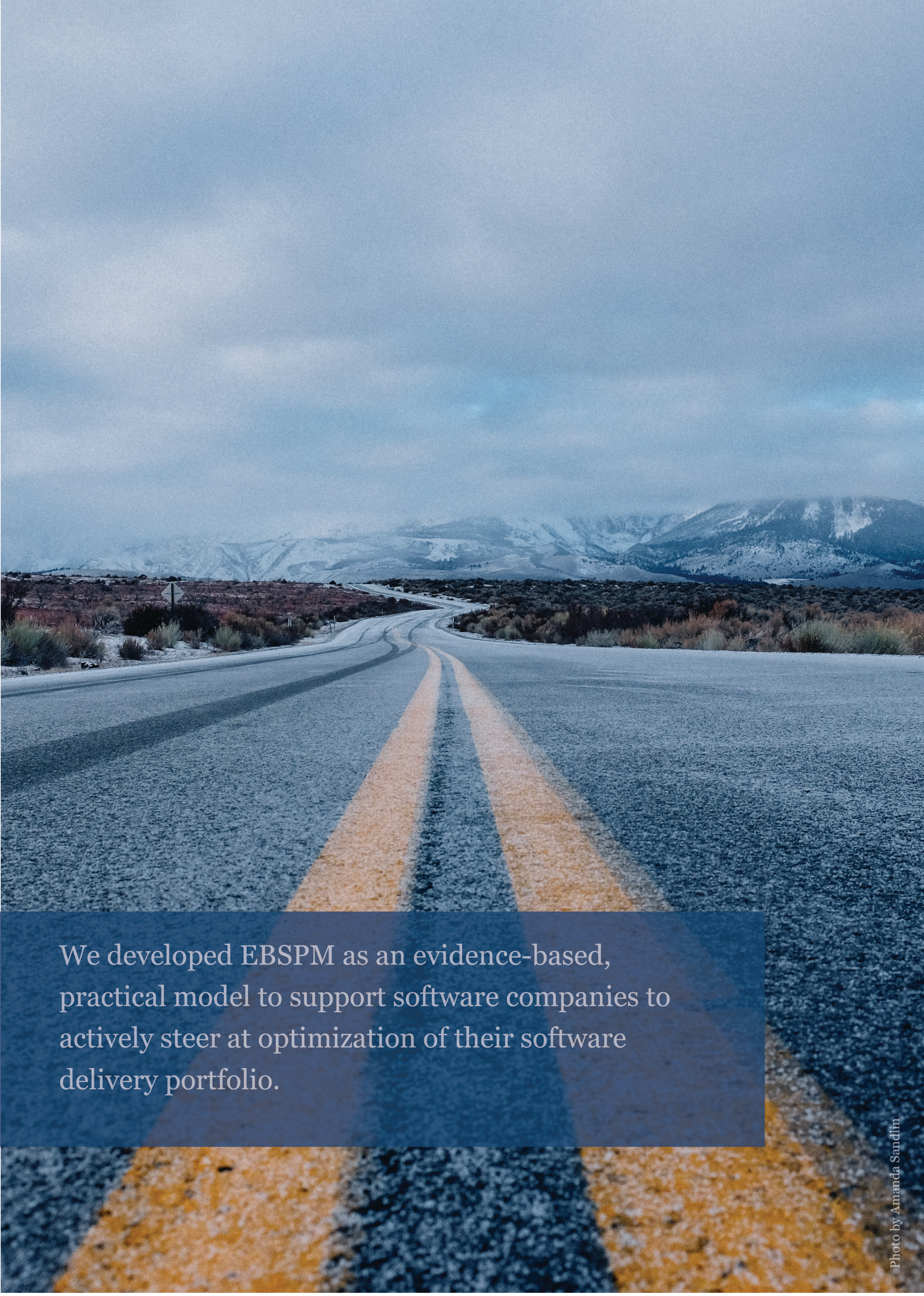
As a result of this ongoing development, and inspired by new and sometimes improved insights into the statistical tests to be used for software analytics, we challenged and improved the tests performed in some chapters.

For this purpose we added an addendum at the end of Chapter 3 in which the results of such improvements are summarized and briefly discussed.

1.4.3. Publications not included in the thesis

Additional publications by the author of this thesis that are not included in this thesis are:

1. *Measuring Best-in-Class Software Releases*. Proceedings of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement (IWSM-MENSURA 2013) (Huijgens & van Solingen, 2013a).
2. *A replicated study on correlating agile team velocity measured in function and story points*. Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics (WETSoM 2014) (Huijgens & van Solingen, 2014a).
3. *An exploratory study on automated derivation of functional size based on code*. Proceedings of the 38th International Conference on Software and Systems Process (ICSSP 2015) (Huijgens, Bruntink, van Deursen, van der Storm, & Vogelezang, 2015b).
4. *Do estimators learn? On the effect of a positively skewed distribution of effort data on software portfolio productivity*. Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics (WETSoM 2016) (Huijgens & Vogelezang, 2016b).
5. *An Exploratory Study on the Effects of Perceived Value and Stakeholder Satisfaction on Software Projects* at the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE 2016) (Huijgens, van Deursen, & van Solingen, 2016c) (best-paper award).
6. *Evidence-based software portfolio management: a tool description and evaluation*. Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE 2016) (Huijgens, 2016a).
7. *Strong Agile Metrics: Mining Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams*. Proceedings of 11th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2017, Industry track) (Huijgens, Lamping, Stevens, Rothengatter, Romano, & Gousios, 2017b).



We developed EBSPM as an evidence-based, practical model to support software companies to actively steer at optimization of their software delivery portfolio.

Photo by Amanda Sandlin

2. A Bird's-eye view on EBSPM

In this chapter, we outline an overall picture of the Evidence-Based Software Portfolio Management (EBSPM) model and the accompanying tool. EBSPM is intended to help software companies in steering their software portfolios based on *cost*, *duration*, and *defects* on the one hand and *quality of estimations*, *stakeholder satisfaction*, and *perceived value* on the other. The research approach is based on instruments such as a *cost duration matrix*, the identification of success and failure factors for software projects, and the collection of data on finalized software projects from portfolios of different companies in a research repository.

This chapter is based on the publications *Evidence-Based Software Portfolio Management* in the proceedings of the doctoral symposium of the 9th International Symposium on Empirical Software Engineering and Measurement (ESEM 2015) (Huijgens, 2015), and *Evidence-based software portfolio management: a tool description and evaluation*. Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE 2016) (Huijgens, 2016). Where applicable adjustments are made to create an overall introduction to EBSPM.

2.1. Introduction

The goal of evidence-based software portfolio management is to use project data collected from the past to predict and monitor the success of other software projects, now and in the future. In such a portfolio management perspective, measuring project size, project costs, project duration and post-release defects is a common practice. Nevertheless, these core metrics only tell a part of the story, and as such companies should be careful in steering their software project portfolios on these data points alone.

It could, after all be possible that a specific project costing twice as much as typical for its size would still be highly valuable to the organization. Performing within time and cost constraints is important, but especially in environments that use agile approaches additional goals enter the arena, such as early delivery of valuable software and an increased focus on stakeholder satisfaction.

Where many other studies use either a quantitative approach (e.g. analyze core metrics) or a qualitative approach (e.g. perform surveys or interviews) to analyze software projects, we combine both ways and look at a company's software project portfolio from a holistic point of view. The goal of our research is to combine a quantitative, data-driven approach on analysis of finalized software project portfolios with a qualitative, survey-based approach to identify factors related to project success and failure, in combination with an approach to measure and analyze *stakeholder satisfaction* and *perceived value* of software projects.

In this chapter, we outline an overall picture of the EBSPM-model and the accompanying tool. To do so, we briefly describe its following main elements as depicted in Table 2.1. In the case studies in the remaining chapters of this thesis the aspects of EBSPM are outlined more in detail.

2.2. The EBSPM-model

The EBSPM-model is an empirical, evidence-based research approach to systematically analyze the software portfolio of software companies, to improve their performance at a company level. EBSPM has been developed in close cooperation with software companies (to be read as information-intensive companies, such as banks, telecom companies, governmental organizations), and it has been set-up in a way that fits as much as possible with practice. Where appropriate case studies are used as the main instrument to address research goals (Runeson et al. 2012) (Yin, 2008). Usually mixed studies are performed: the study includes both *quantitative* and *qualitative* research on the subject projects within a portfolio as a whole of a company or organization. The focus is not to study single software projects as such, but instead to look at the effects of all software projects that were performed over a period

Table 2.1: Overview of the main instruments in the EBSPM-model.

EBSPM Instruments	Description
The EBSPM Approach	An empirical, evidence-based research approach to analyze the performance of a company's software delivery portfolio.
The EBSPM Research Repository	A data set containing data of finalized software deliveries, from four different companies, including Core Metrics, Stakeholder Satisfaction and Perceived Value, and qualitative keywords that characterize deliveries.
The EBSPM Performance Dashboard	Analysis of good practice and bad practice in large, company-wide portfolios of software projects, including a *cost duration matrix* and a summary of Key Performance Indicators.

of time in a portfolio as a whole. By doing so we target both *good practice* projects and *bad practice* projects within a portfolio.

Where applicable electronic surveys among stakeholders of software deliveries are used to collect qualitative data, supplemented with non-structured interviews as techniques to challenge findings from the quantitative analysis.

A precondition that limits the EBSPM-model is the fact that it supports research performed in real, live organizational environments. Therefore the model must not interfere with the daily operation of the studied software projects. Surveys should impose limited burden on people, and analysis should be useful for improvement purposes in daily operations.

In a way the EBSPM-model is set up as an iterative innovation process, or learning cycle. Each iteration consists of collecting data of finalized software deliveries, performing quantitative analysis and benchmarking the results on the EBSPM research repository, and visualizing the outcomes in the EBSPM performance dashboard after completion. The company in scope is assumed to make changes in its software delivery process that might lead to improvements. Based on the outcomes of the analysis new or adjusted research goals are defined, and a new case study starts. In this way such a series of case studies supports continuous innovation of a company's software delivery processes.

2.3. Distinguishing good deliveries from bad ones

The main element within the EBSPM-model is a so-called *cost duration matrix*, that we developed to identify good software deliveries from bad ones. Our premise regarding good and bad deliveries, is that we initially translate success and failure of software projects from its core metrics (Kan, 1995): cost, lead time, and number of defects. To be able to compare different software projects with each other regarding these core metrics, we use functional size (function points) as a normalizer.

2.3.1. Functional Size as a Normalizer

In the EBSPM approach functional size is measured in function points, according to the IFPUG industry standard (IFPUG, 2009). Functional Size Measurement (FSM) is an industry standard to measure size of software engineering activities. It is based on Function Point Analysis (FPA), a method designed by Albrecht in the 70s (Albrecht, 1979), to estimate size of software delivery by means of user functionality. With ISO/ IEC 14143 as an umbrella standard, five FSM methods are certified by ISO as an international standard. Of these five FSM methods the ISO/IEC 20926:2009: IFPUG FSM method (IFPUG, 2009) is used as an industry-wide standard. For this reason we opted for this FSM method as a core metric within EBSPM. The strong positive correlations that are known between functional size on the one hand and cost, duration, and number of defects on the other (Boehm, 1984) (El Emam & Günes Koru, 2008) (Boehm et al., 2000a) (Heemstra & Kusters, 1991) (Bhardwaj & Rana, 2016), gave us the possibility to normalize different software projects through function points. Due to that functional size made it possible to objectively compare individual software deliveries to a larger benchmark.

2.3.2. The Cost Duration Matrix

In Figure 2.2 a single *cost duration matrix* is shown, depicting all projects in the repository. Each project is shown as a circle. The larger the circle, the larger the project is (in function points), and the more red the project is, the more defects per function point it contains. The more blue a project is, the fewer defects per function point it contains. The color range changes from red to blue with light-grey in the middle, indicating the average score for defects

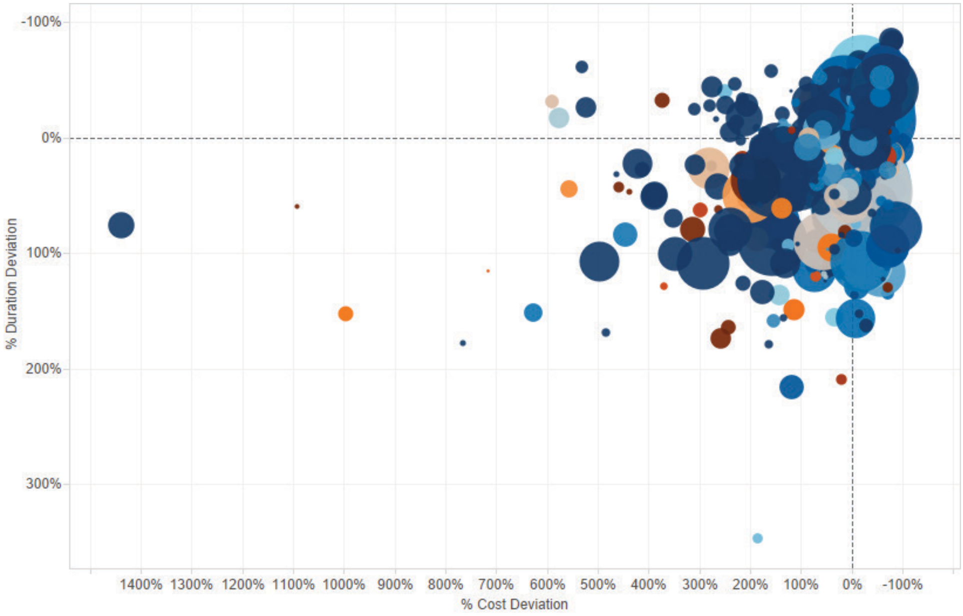


Figure 2.1: The cost duration matrix as the core element in the EBSPM-model.

per function point, measured over the repository as a whole. The position of each project in the matrix represents the cost and duration deviation of the project relative to the benchmark, expressed as percentages. The horizontal and vertical 0%-lines represent zero deviation, i.e. projects that are exactly consistent with the benchmark.

A project at (0%, 0%) would be one that behaves exactly in accordance with the benchmark; a project at (-100%, -100%) would cost nothing and be ready immediately; and a project at (+100%, +100%) would be twice as expensive and takes twice as long as expected from the benchmark.

As Figure 2.1 shows, deviations from the 0%-line on the positive side of both duration and cost (indicating longer durations and higher cost) are huge. The y-axis passes through up to 300%, while the x-axis even extends to more than 1400%. This deviation is mainly caused by a limited number of outliers; without only six outliers the x-axis extends up to 500% deviation.

Figure 2.2 shows the same *cost duration matrix*, with both the x-axis and the y-axis cut-off at the 200%-lines. As can be seen, the projects in the portfolio are divided rather evenly over four different areas, or as we call it

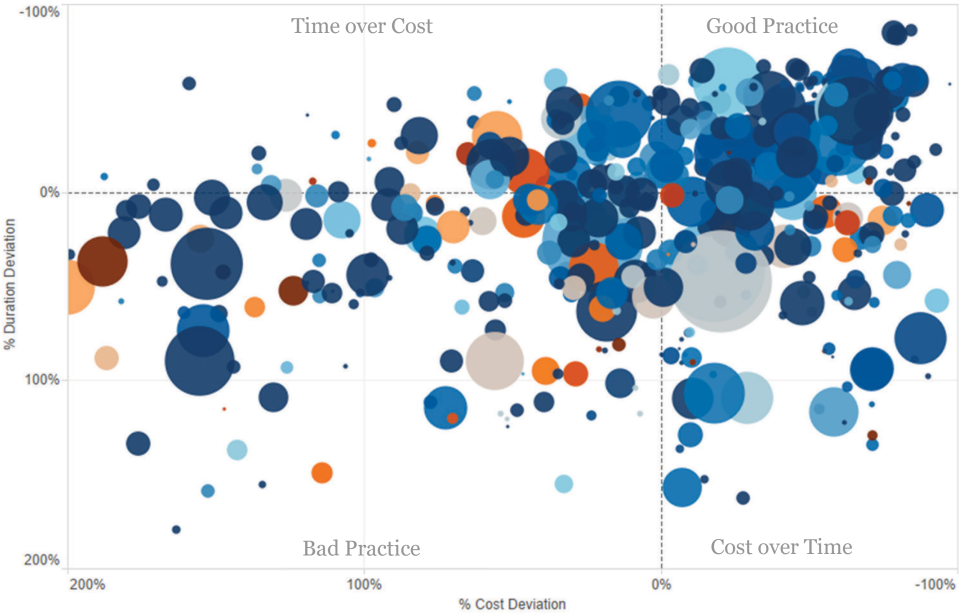


Figure 2.2: The cost duration matrix with cut-off axes.

quadrants. As an example: some are relatively cheaper than the benchmark would predict (right of the vertical 0%-cost bar), yet take longer than expected (below the horizontal 0%-duration bar). The 0%-lines divide the *cost duration matrix* into four quadrants:

1. *Good practice* (top right); projects that score better than average for both cost and duration.
2. *Cost over time* (bottom right); projects that score better than average for cost, yet worse than average for duration.
3. *Bad practice* (bottom left); projects that score worse than average for both cost and duration.
4. *Time over cost* (top left); projects that score better than average for duration, yet worse than average for cost.

Keep in mind that the underlying nominator for all software projects in the *cost duration matrix* is functional size, measured in function points (FPs).

Due to this we can compare the performance in terms of cost, duration, defects found, satisfaction, and value of projects with different sizes with each other.

Based on the EBSPM-model we developed a specific EBSPM-tool. The tool offers two basic features, a research repository and a performance dashboard. These initial features are described in the following Subsections.

2.4. The EBSPM Research Repository

All data that is collected within the scope of the EBSPM-model is stored in an EBSPM research repository, holding the metrics as mentioned in Table 2.2. For a period of seven years we collected performance data of finalized software projects in industry, in close cooperation with a number of large banking and telecom companies in the Netherlands and Belgium. Based on this we built a research repository of core metrics data of more than 500 software projects. In the remainder of this Subsection the metrics mentioned in Table 2.2 are discussed briefly. In the case studies in the remaining chapters of this

Table 2.2: Overview of the main metrics in the EBSPM Research Repository.

EBSPM Metrics	Description
Core Software Delivery Metrics	The four core metrics on software deliveries: size, cost, duration, and number of defects.
Estimation Quality Factor (EQF)	A measure of the deviation of a forecast to the actual cost or duration; a forecasting metric that depicts the quality of forecasts made during a project.
Cost Duration Index	A measure of the relative position of a project within the *Cost Duration Matrix*, represented as a number between zero and one hundred.
Stakeholder Satisfaction	A qualitative measure of the satisfaction of stakeholders of a specific project on the way a project was performed and with the results as delivered.
Perceived Value	A qualitative measure of the perception of stakeholders of a specific project on the amount of value delivered.
Software Delivery Keywords	A series of keywords that characterize a specific software project.

thesis these metrics are outlined more in detail. The EBSPM research repository is available as open source via 4TU Centre for Research Data (Huijgens, 2017a).

2.4.1. The Core Software Delivery Metrics

Within the scope of the EBSPM approach four core metrics are collected for each software delivery in the EBSPM research repository: functional size, cost, duration, and number of defects. The effect on strong positive correlations between these metrics is well known from related work (Huijgens et al., 2014b) (Boehm, 1984) (El Emam & Günes Koru, 2008) (Boehm et al., 2000a) (Heemstra & Kusters, 1991) (Bhardwaj & Rana, 2016). Also the effect of functional size as a risk factor has been described earlier. Smaller projects tend to have lower cancellation rates (Rubinstein, 2007) (Sauer & Cuthbertson, 2003). Smaller projects tend to perform better in terms of quality, being on budget, and being on schedule (Rubinstein, 2007) (Sauer & Cuthbertson, 2003) (Sonnekus & Labuschagne, 2004). Project size is found to be an important risk factor for success (Barki, Rivard, & Talbot, 1993) (Jiang & Klein, 2000) (Schmidt, Lyytinen, Cule , & Keil, 2001) (Zowghi & Nurmuliani., 2002) (Heemstra & Kusters, 1989) (Chidambara & Senthil Kumar, 2016).

Based on these four core metrics, three key performance indicators are calculated: *cost per function point*, *days per function point*, and *defects per function point*, using in each case the size in function points as weighting factor.

A limitation with regard to the EBSPM-model is that in practice collection of effort data of finalized software deliveries is not mandatory. Experience in industry taught us that, especially in outsourcing cases, reliable effort data is difficult to measure, if not impossible in many cases. In Chapter 6 of this thesis we focus more in depth on the relation between cost and effort and on the backgrounds of collecting both metrics.

2.4.2. Estimation Quality Factor

The *estimation quality factor* (EQF) is a measure of the deviation of a forecast to the actual cost or duration. EQF is a forecasting metric that depicts the quality of forecasts made during a project. The measure was defined by DeMarco (1984). He defines EQF by:

$$EQF = \frac{Area\ under\ actual\ value}{Area\ between\ forecast\ and\ actual\ value}$$

We use the formulization proposed by Eveleens & Verhoef (2009). EQF allows us to quantify the quality of forecasts. A low EQF value means that the deviation of the forecasts to the actual cost or duration is large. EQF is measured for both cost and duration. In Chapter 7 of this thesis we elaborate the specific application of EQF for both cost and duration, and the correlations with other metrics more in detail.

2.4.3. The Cost Duration Index

The *cost duration index* is a measure of the relative position of a project within the *cost duration matrix* (see Subsection 2.5.2). The index is represented as a number between zero and one hundred. In practice most projects score between 80 and 99. A high index corresponds to a good position in the *cost duration matrix*. The index is based on the geometric mean of two proportions comparing the actual value to the benchmark value. In Chapter 7 of this thesis the concept of *cost duration index*, including any correlations with other metrics, is elaborated more in detail.

2.4.4. Stakeholder Satisfaction

Stakeholder satisfaction is a measure of the satisfaction of stakeholders of a specific project with the way a project was performed and with the results as delivered by that project. *Stakeholder satisfaction* is measured by asking stakeholders of a specific project to rate their satisfaction on two aspects; the way a project was performed (the project's process), and with the results as delivered by a project (the project's result), for which we use questions with a 1 to 5 rating scale. We use electronic surveys to collect data on *stakeholder satisfaction*. Surveys are sent after finalization of each software delivery to all internal, and if applicable all external, stakeholders of deliveries: e.g. project managers, developers, testers, product owners. *Stakeholder satisfaction* is in our approach not weighted amongst stakeholders. In Chapter 7 of this thesis the concept of *stakeholder satisfaction*, including any correlations with other metrics, is elaborated more in detail.

2.4.5. Perceived Value

Value of software projects is a complex metric to measure (Shepperd, 2014), and studies are not specific on how they define value (Dingsøyr & Lassenius, 2016). It is difficult, if not impossible, to measure objectively and indisputably the real value as delivered by software projects to customers of software companies. Is *real value* about money and time as Beck says (Beck, 2000)?

Does it mean financial value, as in studies indicated by *return on investment* (ROI) (Solingen, 2004)? Or is *real value* measured by *net promotor score* (NPS), as other studies indicate (Green, 2011) (Hofner, Mani, Nambiar, & Apte, 2011) (Feyh & Petersen, 2013)? Such holistic measurements on value are often difficult to make for a single project, and they cannot easily be related to single software projects, mainly because too many different factors are of influence for such measurements.

To approach the real value, we measure *perceived value* as a quantitative measure of the perception of stakeholders of each project. This is based on the notion that in fact every measurement is an agreement on a measurement procedure that sufficiently approaches the actual value (Solingen, 2004). We measure *perceived value* alike *stakeholder satisfaction* in an electronic survey among software delivery stakeholders. *Perceived value* is measured for each stakeholder in a specific delivery, on four aspects: a company's customers, a company's financials, a company's internal process effectiveness, and a company's innovation.

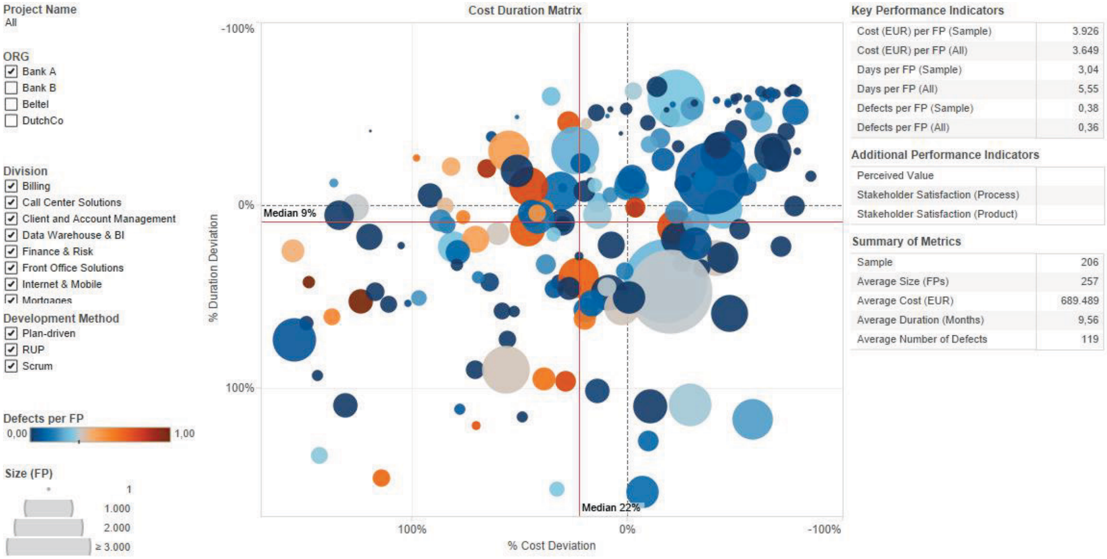
We base the use of the four perspectives *customer*, *financial*, *internal process*, and *innovation* on the Balanced Scorecard (Kaplan & Norton, 1995). Based on the results per project of the four *perceived value* measures a *perceived value (overall)* is calculated, with the number of measures as weighting factor (since answering each separate question on *perceived value* is not mandatory, only in case a stakeholder mentions a value this is incorporated in the calculation of an overall value, not counting the choice "Don't know").

Chapter 7 of this thesis gives a detailed view on *perceived value*, including any correlations with other metrics.

2.4.6. Software Delivery Keywords

Besides the set of metrics described above the EBSPM research repository contains a variety of keywords that characterize a software delivery in a specific way. Multiple keywords can be mapped on one delivery. When preparing the overview of keywords we used a practice-driven approach to identify those keywords, that were available within the different software companies that were in scope of our research. We specifically looked at project characteristics, reasons to start a project, release-based way of working, delivery approach specifics, and how teams were organized. The following keywords are applicable:

1. Dependencies
   1.1. Single-application;
   1.2. Phased project (part of program);
   1.3. Dependencies with other systems;
2. Reason behind project
   2.1. Business driven;
   2.2. Technology driven;
   2.3. Rules & Regulations driven;
   2.4. New technology, framework solution.
3. Project Specifics
   3.1. Migration;
   3.2. Legacy;
   3.3. Security;
   3.4. Pilot; Proof of Concept;
   3.5. Bad relation with external supplier;
4. Package Solution
   4.1. Package off-the-shelf;
   4.2. Package with customization;
5. Team Specifics
   5.1. Once-only project;
   5.2. Fixed, experienced team;
   5.3. Many team changes, inexperienced team;
6. Release Characterization
   6.1. Release-based, mapped on one application;
   6.2. Multi-application release;



Project Name
All

ORG
☑ Bank A
☐ Bank B
☐ Beltel
☐ DutchCo

Division
☑ Billing
☑ Call Center Solutions
☑ Client and Account Management
☑ Data Warehouse & BI
☑ Finance & Risk
☑ Front Office Solutions
☑ Internet & Mobile
☑ Mortgages

Development Method
☑ Plan-driven
☑ RUP
☑ Scrum

Key Performance Indicators	
Cost (EUR) per FP (Sample)	3.926
Cost (EUR) per FP (All)	3.649
Days per FP (Sample)	3,04
Days per FP (All)	5,55
Defects per FP (Sample)	0,38
Defects per FP (All)	0,36

Additional Performance Indicators	
Perceived Value	
Stakeholder Satisfaction (Process)	
Stakeholder Satisfaction (Product)	

Summary of Metrics	
Sample	206
Average Size (FPs)	257
Average Cost (EUR)	689.489
Average Duration (Months)	9,56
Average Number of Defects	119

Figure 2.3: An example of the EBSPM performance dashboard, with a selection of software deliveries in Bank-A (a large Dutch bank).

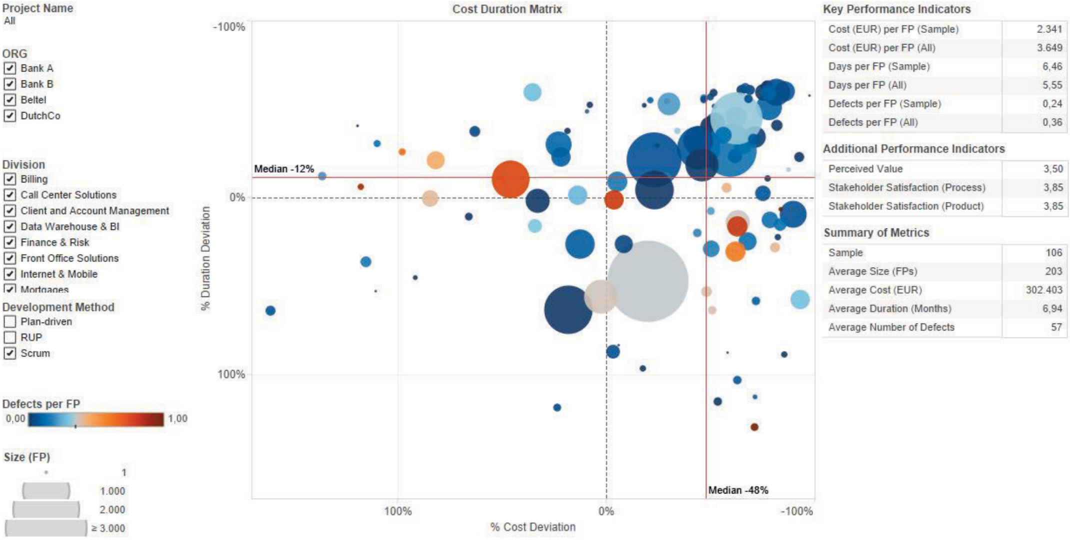
7. Delivery Approach
   7.1. Steady heartbeat;

2.5. The EBSPM Performance Dashboard

The main element within the EBSPM-model to visualize the outcomes of analysis on data from the EBSPM research repository is the EBSPM performance dashboard. Figure 2.3 gives an overview of the dashboard, with a selection made on projects only delivered in *Bank-A* (a large bank in the Netherlands). In Figure 2.4 the dashboard is depicted with a selection of deliveries that were developed in an agile (Scrum) way. As both figures show, three parts can be distinguished in the dashboard:

1. On the left side a number of selection options;
2. In the center a *cost duration matrix*;
3. At the right side a summary of key performance indicators.

These three parts of the dashboard are described more in detail in the following Subsections.



Project Name
All

ORG
☑ Bank A
☑ Bank B
☑ Beltel
☑ DutchCo

Division
☑ Billing
☑ Call Center Solutions
☑ Client and Account Management
☑ Data Warehouse & BI
☑ Finance & Risk
☑ Front Office Solutions
☑ Internet & Mobile
☑ Mortgages

Development Method
☐ Plan-driven
☐ RUP
☑ Scrum

Key Performance Indicators

Cost (EUR) per FP (Sample)	2.341
Cost (EUR) per FP (All)	3.649
Days per FP (Sample)	6,46
Days per FP (All)	5,55
Defects per FP (Sample)	0,24
Defects per FP (All)	0,36

Additional Performance Indicators

Perceived Value	3,50
Stakeholder Satisfaction (Process)	3,85
Stakeholder Satisfaction (Product)	3,85

Summary of Metrics

Sample	106
Average Size (FPs)	203
Average Cost (EUR)	302.403
Average Duration (Months)	6,94
Average Number of Defects	57

Figure 2.4: An example of the EBSPM performance dashboard, with a selection of software deliveries that were developed in an agile (Scrum) way.

2.5.1. Selection Options

The EBSPM performance dashboard offers four (standard) selection options:

1. *Organization*: the repository holds data of four different software companies, two of which are a large bank in the Netherlands, one medium-sized Belgian telecom company, and one small software company in The Netherlands that offers Billing solutions to telecom companies in Europe.
2. *Business domain*: the repository contains data from several business domains, such as Internet, Mobile, Payments, Mortgage, Data Warehouse & BI, and Call Center Solutions).
3. *Development method*: the repository holds data that is collected from projects with different development methods, such as plan-driven (waterfall), RUP, and agile (Scrum).
4. *Project name*: a selection can be made based on a project name or a part of such a name.

Once a selection is made, the subset of projects in the repository within the selection is shown in the *cost duration matrix*, and included in the calculation of key performance indicators. The overall performance of the portfolio is furthermore summarized through the two red 'median' lines.

2.5.2. The Cost Duration Matrix as the Core of the Dashboard

The core of the EBSPM performance dashboard is the *cost duration matrix*, that we developed to compare a (selected) subset of projects to the benchmark (the content of the EBSPM research repository as a whole) (Huijgens et al., 2014c) (Huijgens et al., 2015c) (see Subsection 2.3.2 for a detailed explanation of the *cost duration matrix*).

2.5.3. The Key Performance Indicator Summary

At the right of the EBSPM performance dashboard (see Figure 2.3) a number of key performance indicators, some additional performance indicators, and summary metrics are indicated. At the top *cost per FP*, *days per FP*, and *number of defects per FP* are shown for both the EBSPM research repository as a whole, and the selected subset.

As an example: Figure 2.3 shows that the *cost per FP* calculated over the repository as a whole is 3636 Euro, while the *cost per FP* for the selected subset of *Bank-A* is 3926 Euro. Keep in mind that we used *project size* as a weighting factor, instead of number of projects.

In the right center part three metrics are shown that indicate *stakeholder satisfaction* and *perceived value*. *Stakeholder satisfaction* is shown for two aspects: *process* and *result* (the delivered product).

Below that, the sample size of both the repository as a whole, and the selected subset are shown, and a number of average values for *project size*, *project cost*, *project duration*, and *number of defects*.

2.5.4. Who should use the tool? And why?

The goal of the users of the EBSPM-tool is to twofold. Firstly, the tool enables them to continuously analyze the performance of their software delivery processes in terms of time, number of defects, and money. Selections for over-time analysis are easily to be added, and the user cab drill down to business domain level, organization level, or development method. The tool helps users to understand questions such as:

"Is the performance of software deliveries in the payment domain of my company improving over time?"

"Is it wise to use Scrum as a development approach? Or should I stick to a traditional, plan-driven way of developing software?"

A second goal for the users of the EBSPM-tool is to use it as a source for estimating new projects. The tool helps to understand performances, and especially the bandwidths amongst projects within equal business domains, development approaches, and other keywords. It helps users to understand questions such as:

"Should this new project be estimated at a cost of 3 million euro, or is 1 million euro more realistic?"

"My supplier tells me that this project is going to cost me 500 K euro. Is that a fair estimate?"

"The project manager is very satisfied because he finalized the project on time and within the budget... Yet, the customers are very dissatisfied because of many defects, and too high costs. Who is right?"

An evidence-based approach, such as EBSPM, might help software companies to better understand their performances, and to answer questions like the ones stated above.

2.6. A practical, evidence-based approach

We developed the EBSPM-model as an evidence-based, practical set of tools to support software companies to actively steer at optimization of their software delivery portfolio. In the next chapter we describe an EBSPM-tool that we developed to support application of the model in a practical context.

In the Chapters 3 to 7 of this thesis we elaborate on a number of case studies, surveys, and data analysis studies in which we evaluated the EBSPM-model in real cases in the software industry. We studied both large software companies and smaller ones. In a large Dutch bank we collected cost, duration and defects data from software projects that finalized over a period of five years, in order to understand what made projects successful, and what leads to bad performances. Besides that we looked at the quality of project estimations for both cost and duration.

In a medium sized telecom company in Belgium we studied data from finalized projects over a period of three years. We expanded our approach with additional metrics such as *stakeholder satisfaction* and *perceived value*, in order to understand how these related to cost, duration, number of defects and quality of project estimations. Besides that we studied in this company the effects of a statistics-driven, empirical way to define fixed-price project proposals in close cooperation with an Indian supplier that had a strategic, long-term partnership with the Belgian telecom company.

Finally, we examined whether the effects that we found in both the banking and the telecom company, also applied to a much smaller Netherland's-based software company that delivers Billing solutions to telecom operators in Europe. In this company the actual development activities where outsourced to a Dutch software company with development teams in India.

To give an impression of the way the EBSPM-approach matured: the EBSPM research repository grew to more than 500 finalized software projects of which data was collected. The repository represents a total of 291 million Euros that was spent by four companies on development and maintenance of their software projects.

Where the original EBSPM performance dashboard was built in MS Excel, we developed a renewed dashboard by using the business intelligence solution Tableau, offering us better functionality for drill down to lower levels in a company's software project portfolio.

As we described, the EBSPM-model is built on the collection, analysis, and benchmarking of a limited subset of metrics such as *time*, *cost*, *defects*, *functional size*, *estimation quality*, *stakeholder satisfaction*, and *perceived value*. An effect of this could be that the approach offers practitioners and researchers a somewhat limited view at the reality of a company's software portfolio. Other aspects are simply not in scope, while they might be of great importance to a company's performance. Without compromising on the quality of the following chapters, we recommend that those who read the remaining chapters of this thesis should keep this limitation in mind.

Photo by Jon Tyson

We analyzed - from a portfolio point of view - the characteristics of best performers and worst performers, in a dataset of 352 software projects, resulting in 7 success factors and 9 failure factors.

3. On Good Practice and Bad Practice

Context: What can we learn from historic data that is collected in three software companies that on a daily basis had to cope with highly complex project portfolios? *Objective:* In this chapter we analyze a large dataset, containing 352 finalized software engineering projects, with the goal to discover what factors affect software project performance, and what actions can be taken to increase project performance when building a software project portfolio. *Method:* The software projects were classified in four quadrants of a *cost duration matrix*: analysis was performed on factors that were strongly related to two of those quadrants, *good practice* and *bad practice*. A ranking was performed on the factors based on statistical significance. *Results:* The chapter results in an inventory of 'what factors should be embraced when building a project portfolio?' (success factors), and 'what factors should be avoided when doing so?' (failure factors). *Conclusions:* The major contribution of this chapter is that it analyzes characteristics of best performers and worst performers in the dataset of software projects, resulting in 7 success factors, and 9 failure factors.

This chapter was published as *How to build a good practice software project portfolio?* in the 2014 companion proceedings of the 36th International Conference on Software Engineering (ICSE SEIP 2014) (Huijgens, van Solingen, & van Deursen, 2014).

3.1. Introduction

Growing complexity faces many software engineering companies nowadays with a portfolio- and project control capability that is lagging behind with their high IT-expenditure. A trend towards rapid application development, acceleration of the pace of change in information technology, in organizations, in competitive countermeasures, and in the environment has caused increasing frustration with heavyweight plans (Boehm, 2006a). An answer to this challenge can be found in a need for instruments and techniques that support transparency and orchestration of software engineering activities, showing organizations what they can learn from their best performing projects (*good practice*), and from their worst performing projects (*bad practice*). Especially in complex environments successful software engineering requires companies to pay special attention to a learning capability that supports flexible portfolio- and project management to prevent from decreasing productivity, increasing time-to-market, and low software quality (Tihinen, Parviainen, Suomalainen, & Karhu, 2011).

However this problem seems hard to solve. Still many software development organizations have enormous difficulties developing reliable effort estimates that result in on-time and on-budget delivery of their software products, (Moløkken & Jørgensen, 2003). Many companies have no or limited historic reference data on their software engineering activities available (Dagnino, 2013). This limits existing solutions on software estimation to immature and unreliable estimation techniques and hinders learning. An often-heard goal of continuous improvement seems searching for the pot of gold at the end of the rainbow.

What can we learn here from historic data that is collected in three software engineering companies that on a daily basis had to cope with such highly complex project portfolios?

3.1.1. Research Objectives

Within the scope of the implementation of several measurement programs as part of process improvements, a large dataset containing 352 software projects was collected in practice in three different companies during a time-span of six years (2008 to 2013). We define the following research question:

RQ: What factors affect software project performance, and what actions can be taken to increase project performance when building a software project portfolio?

3.1.2. Context

Data was collected on finalized software engineering projects within three different companies. The companies – two large banks (in this chapter referred to as '*Bank-A*' and '*Bank-B*'), and one telecom provider (referred to as '*BelTel*') – were, with regard to their software engineering activities, comparable to each other. The size of the software project portfolio of the banks was considerably larger than that of the *BelTel*: The measured yearly throughput of software projects for *Bank-A* was approximately 10,000 function points (FPs) (IFPUG, 2009) (NESMA, 2004), for *Bank-B* this was approx. 8,000 FPs per year, and *BelTel* measured approx. 2,000 FPs per year. However, on a business domain scale the software engineering activities were equal in many ways. Software engineering was organized in projects or releases, different delivery models were adopted, and software engineering was characterized by a variety of programming languages and business domains (varying from business intelligence systems to mobile apps).

In all three companies a comparable and similar approach for measurement and analysis of software projects was implemented during the data collection period. A so-called *learning-cycle* was implemented: (1) finalized software projects were measured, collected in a measurement repository, and analyzed, (2) data of groups of finalized projects was analyzed on specific trends and benchmarked with internal and external peer-groups, and (3) finally trends were incorporated in an estimation process for newly to be started projects. With regard to step 1 of the learning cycle the measurement repository included both quantitative project data (e.g. size, cost, effort, duration, and defects) and qualitative project data (e.g. reasons named by the project manager that could explain the projects' performance).

In this chapter we subsequently discuss research design, execution, analysis, interpretation, related work, and conclusions and future work.

3.2. Research Design

3.2.1. Approach

There may be a large gap between how different stakeholders, and researchers, define success and failure of software projects. Similarly to (Lindberg, 1999), we relate success and failure within this research to better or worse than average cost, effort, and duration performance. Although, where (Lindberg, 1999) compares to industry, we posed to focus the study on identifying success and failure factors of the sample software projects themselves. Every single project from the sample is compared with the average performance of the whole repository. The idea behind this is that a focus at identifying projects that performed better than average, and projects that performed worse than average, might help companies to realize their ultimate goal of continuous improvement. The refined research objectives of this study are to identify factors that are instrumental in software project success or failure. Following from that, we analyze actions that help to improve the performance of such projects. In the scope of this study a performance-rating 'better than others' must be read as better than the average performance of the whole sample. 'Worse than others' must be read as worse than the average performance of the whole sample.

3.2.2. Design

The strategy for this research can be defined as a data analysis project. We did not study individual projects in our sample, but the primary author collected and maintained the majority of the projects in the measurement repository. A minority of the projects was collected by third parties. Our goal was to explore the performance of finalized software projects and identify key success and failure factors in order to build a *good practice project portfolio*. Our study proposition was twofold. First, we hypothesized that within the repository good performers and bad performers can be distinguished in a quantitative way. Second, we expected that the qualitative information in the repository gives valuable, additional information on success and failure of the projects and on software engineering projects in general.

3.2.3. The Research Repository

Table 3.1 gives an overview of the research repository, including the different aspects that are analyzed in our research. All data in the repository was collected before we started our research, and only with practical analysis purposes in mind. All data is about finalized projects: the dataset does not hold any data on prematurely stopped or failed projects. All projects are related to solution delivery, i.e. an information system was modified or completely new designed. The dataset contains software engineering projects, which in some cases contain an infrastructure component or an implementation of middleware: no full infrastructure or middleware projects are included in the repository. With regard to our research, the dataset is where possible and applicable, discussed with the measurement team that was responsible for the collection. For the research described in this Section all available data in the repository has been used; no deviations or outliers have been removed from the dataset.

The repository contains both quantitative data (e.g. ratios on size, duration, and cost), and qualitative data (e.g. applicable keywords). This data in particular was collected due to its availability in the software companies where research was performed. Due to these practical limitations for example only a limited number of projects in the repository have effort data available, since most companies involved did not collect (actual and estimated) effort data of software projects.

The repository holds data of 352 software engineering projects carried out in three companies during a period from 2008 to 2013. The projects represent a total amount spent of 266M Euro. Project cost range from 12K Euro to 6.8M Euro. The sum of function points across projects in the repository is 91,105 FPs; ranging from projects with a size of 5 FPs to 4,600 FPs. The project duration ranges from 0.9 Months to 26.8 Months.

Qualitative data is recorded for the following research categories:

1. Business domain (BD);
2. Primary programming language (PPL);
3. Company (ORG);
4. Delivery model (DM);
5. Development class (DC);
6. Size category (SC), and;
7. Project keyword (PK).

Within these 7 research categories are in total 56 project factors inventoried in the repository (see Table 3.1 for an overview).

For all inventoried projects size is measured in Function Points (FPs). Function Point Analysis (FPA) has been performed either by an expert member of a measurement team or an external certified FPA-specialist, according to ISO-standardized functional size measurement (FSM) methods (NESMA, 2004) (IFPUG, 2009).

Table 3.1. Overview of the EBSPM Research Repository

Category	Type	Occurrence	N	Definition of Project Factors
Company ID (ORG)	Nominal	3	352	Identification code of the company where a project was performed; three companies were applicable (nr. of occurrence between brackets): Bank A (206), *Bank-B* (125), BelTel (23).
Project ID	Nominal	352	352	Identification code of a project.
Year of Go Live	Ordinal	6	352	Year when a project was finalized; the following years Go Live were applicable: 2008 (32), 2009 (59), 2010 (81), 2011 (131), 2012 (41), 2013 (10).
Business Domain (BD)	Nominal	10	352	Customers business sector; the following BD were applicable: Finance & Risk (54), Internet & Mobile (54), Payments (50), Client & Account Management (incl. CRM systems) (46), Savings & Loans (40), Organization (incl. HRM) (31), Call Centre Solutions (21), Mortgages (21), Data warehouse & BI (18), Front Office Solutions (17).
Primary Programming Language (PPL)	Nominal	21	352	Primary used programming language: JAVA (154), .NET (59), COBOL (55), ORACLE (29), SQL (9), 3GL (8, unknown was what specific languages were applicable here), Visual Basic (6), RPG (6), FOCUS (5), PowerBuilder (5), PRISMA (4), MAESTRO (3). In the analysis 4th Generation (1), PL1 (1), JSP (1), C++ (1), Clipper (1), Document (1), PL/SQL (1), Siebel (1) and Package (1, unknown what specific language was applicable) were referred at as Other.
Delivery Model (DM)	Nominal	2	352	Classification of the used delivery model; two DM were applicable: Structured (e.g. Waterfall) (307), and Agile (Scrum) (45). One project reported as DM RUP is included in the analysis of Structured.
Development Class (DC)	Nominal	4	352	Classification of the development: New development (173), Major enhancement (25-75% new) (124), Minor enhancement (5-25% new) (27), Conversion (28).

Category continued	Type	Occurrence	N	Definition of Project Factors
Project Keyword (KW)	Nominal	20	351	Characteristics on a specific project (multiple keywords could be mapped on one project, on one project no keyword was mapped); the following keywords were applicable: Single-application (270), Business driven (150), Release-based (one application) (144), Once-only project (122), Phased project (part of program) (65), Fixed, experienced team (62), Technology driven (58), Steady heartbeat (49), Dependencies with other systems (41), Migration (35), Rules & Regulations driven (33), Multi-application release (21), Many team changes, inexperienced team (17), Package with customization (16), Legacy (15), Security (14), Pilot; Proof of Concept (10), Bad relation with external supplier (9), New technology, framework solution (3), Package off-the-shelf (1).
Size (FP)	Ratio	-	352	Size of a project in Function Points (FPs).
Duration	Ratio	-	352	Duration of a project in Months; measured from the start of Project Initiation to (technical) Go Live.
Cost	Ratio	-	352	Cost of a project in Euros; measured from the start of Project Initiation to (technical) Go Live.
Effort	Ratio	-	352	Effort spent in a project in Person Hours (PHRs); measured from the start of Project Initiation to (technical) Go Live.
Defects	Ratio	-	172	The number of errors or faults found in a project from System Integration Test to (technical) Go Live. Not for all projects defects were administrated; for 172 projects defects info was recorded in the repository.

*No occurrences are indicated for the 5 last measures in the inventory, since these are different for every measured project.

FPA was performed based on sets of final project documentation that usually were delivered by the project manager. All projects were collected and analyzed according to the so-called SEI Core Metrics, a standard set of process-based software engineering metrics (CMMI Product Team, 2010) (Kan, 1995). Project duration was measured in a number of months from the start of the project initiation to technical Go Live.

Effort was measured in hours. Although measurement specialists that were responsible for the data collection reported that data quality with regard to effort was low, especially where (globally distributed) external suppliers were involved in a project. Because of that we decided to use both effort and cost data for analyzing purposes. Project cost – in this case all project related cost excluding investments (e.g. software license costs) – were recorded in euros in the measurement repository. Besides that for a limited set of projects (N = 172) the number of defects was recorded: all pre-release bugs available in the project administrations where included, post-release incidents where not included.

3.2.4. Analysis Procedure

All software projects in the measurement repository are analyzed from a portfolio point of view, meaning that we were interested in particular in the mutual coherence between the measured projects. We perform the analysis of the portfolio (the measurement repository) in four steps:

First, we analyze the overall average performance of all projects in the repository with regard to *project size*, *project cost*, and *project duration*, measured in function points, euros, and months, respectively.

Subsequently, we analyze what projects performed as a *good practice*, and what projects did perform as a *bad practice*. In the context of our research a project is assessed as a *good practice* when the performance on both cost and duration is better than the average cost and duration corrected for the applicable project size. A project is assessed as a *bad practice* when the performance on both cost and duration is worse than the average cost and duration again correct for the project size. To do so we classify the projects in a *cost duration matrix* (see Figure 3.2). We challenge the project performance exclusively against the performance of the whole sample (internal benchmarking). We compare the outcome of our analysis with other studies in Section 3.7 on Related Work.

Once all projects are classified in the *cost duration matrix*, we analyze how the 56 project factors (as defined in Table 3.1) are related to the four quadrants of the matrix. This analysis results in a percentage based on number of projects per quadrant for every project factor and a percentage based on cost per quadrant for every factor. We assess the outcome of the analysis by calculating the significance of both outcomes (a range of percentages per quadrant based on number of projects, versus a range of percentages per quadrant based on project cost), by performing a chi-square test.

An example: for the Primary programming Language (PPL) ORACLE 29 projects are measured, of which 26 small releases (measured in cost and size) score as a *good practice*. Although, when assessed based on project cost it shows that 3 large projects (in size and cost) score as a *bad practice*. This leads to an indistinct result that from a number of project point of view PPL ORACLE scores high in *good practice*, and from a cost point of view it scores high in *bad practice*. In the further analysis factors with these kinds of indistinct outcomes will be excluded.

Finally, we identify factors that are strongly related for the composition of software engineering project portfolios, by analyzing specific subsets per research aspect. In other words: ‘What factors should be embraced when composing a project portfolio?’ and ‘What factors should be avoided when doing so?’ For this analysis we define a research aspect to be ‘strongly related’ when the percentage *good practice* or *bad practice* was 50% or more.

Once an inventory of strongly related factors is finalized we test for each of them whether it indeed affects the probability that a project ends up as a *good* or *bad practice*. To that end, for each strongly related factor F, we generate a null and alternative hypothesis:

- H(F, 0): Factor F does not influence good (or bad) practice;
- H(F, 1): Factor F influences good (or bad) practice.

To test these hypotheses, we use the binomial distribution to estimate the chance that our actual observation takes place under the null hypothesis, as follows.

Let n be the total number of projects in the repository. Let *prob(F, n)* be the proportion of projects for which *F* holds in *n*, and let *k* be the number of good (bad) practice projects in which *F* holds. Then *p* is given by the binomial distribution as follows:

$$p = (n\ over\ k) * prob(F,n)^\wedge k(1 - prob(F,n))^\wedge(n-k)$$

We reject the null hypothesis in favor of the alternative hypothesis if the significance p is below 0.05.

3.3. Execution

3.3.1. Distribution of the Sample

The distribution of the dataset is described as a positively skewed distribution, or one whose elongated tail extends to the right end of the range. The mean duration in the sample is 8.7 Months, while the median duration is 8.0 Months (min 0.9 Months, max 26.8 Months). The mean project cost is EUR 750,777, while the median project cost is EUR 459,150 (min EUR 12,233, max EUR 6,802,466). The mean project size is 259 FPs; the median project size is 147 FPs (min 5 FPs, max 4,600 FPs).

Although due to the positively skewed distribution analysis based on the median (instead of the mean) might be preferable (the expectation is that outliers will interfere the outcome less), we use the mean in our analysis of factors. After performing both analyses we found, except for some small differences in numbers and percentages, no differences in the outcome of the study (success and failure factors). We assume the Central Limit Theorem to be applicable in this case.

3.4. Analysis

3.4.1. Overall Performance Analysis

We analyze the overall weighted average performance, with project size (FPs) as weighting factor, of all projects in the repository with regard to project size, project cost, and project duration. For this purpose we calculate three performance indicators:

1. *Cost per function point*: expressed in cost per size unit (Euro/FP) and size unit per hour (FP/HR);
2. *Duration per function point*: expressed in project calendar days per size unit (Days/FP);

Table 3.2. Average Performance per Size Category.

Performance Indicator	SP	SMP	LMP	LP	Overall
Cost per FP (Euro/FP)	4,364	3,395	2,508	2,111	2,929
FP per Hour (FP/HR)	0.024	0.034	0.045	0.047	0.037
Duration per FP (Days/FP)	2.74	1.03	0.75	0.38	1.08
Number of Defects per FP	0.20	0.13	0.13	0.21	0.18
Percentage in Sample	62%	19%	10%	9%	100%

Explanation of abbreviations: SP = Small Projects (<200 FP); SMP = Small Medium Projects (201-400 FP); LMP = Large Medium Projects (401-600 FP); LP = Large Projects (>601 FP); PROD = Productivity in resp. Euros per FP and FP per hour; TTM = Time-to-Market in Days per FP; PQ = Process Quality in Defects/ FP. The indicators are calculated as weighted average, with Size as weighting factor (e.g. Cost (Euros) divided by Size (FP), instead of number of projects as weighting factor.

3. *Number of defects per function point*: expressed in quality of the process per size unit (Defects/FP).

In Table 3.2 the values of these indicators are inventoried. In order to get more insight in the effects of economy of scale, we divided the sample in four size categories. The used bins, size categories of 200 FP each, all projects larger than 600 FPs are considered large, are commonly used as a measure for project size in two of the applicable companies. Table 3.2 gives an overview of the weighted average scores per size category within the measurement repository with regard to the performance indicators.

As the table shows economy of scale does play an important role here: the performance of the projects in the repository, measured in *cost per FP*, *duration per FP*, and *number of defects per FP*, is related to the size of a project. The larger the project; the better the performance on average is in terms of time, money, and quality. The table further shows that most projects in the repository could be categorized as small projects (62% of the projects are smaller than 200 FP).

The analysis shows a remarkable fact: while on the one hand medium sized projects show the best performance in terms of time, money, and quality, on the other hand companies give preference to build their portfolio on small or small medium sized projects. An often witnessed adage that ‘small projects

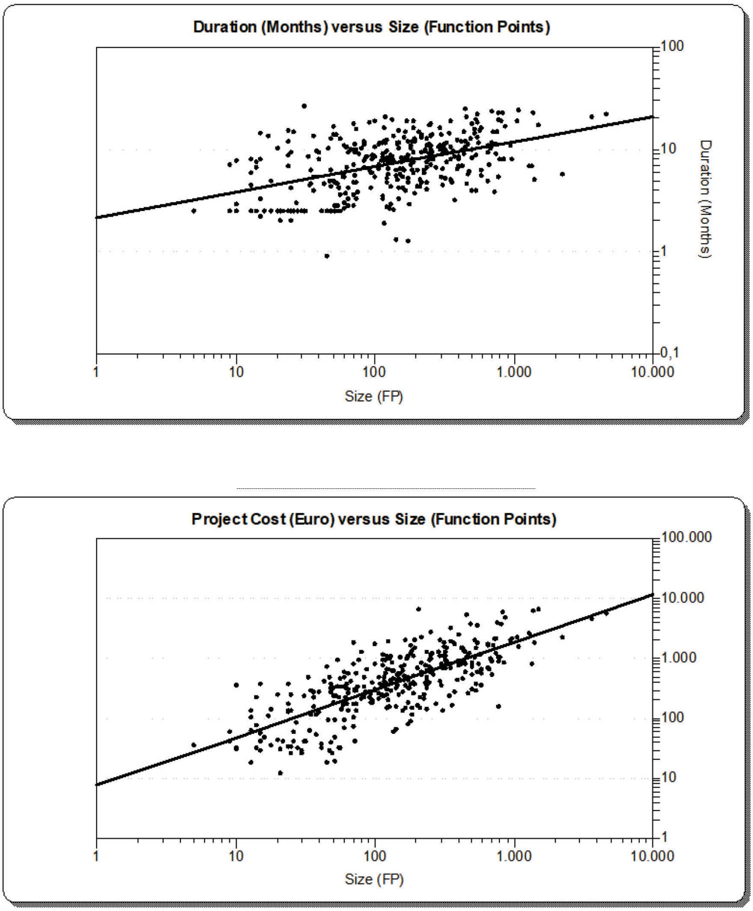


Figure 3.1: Two plotter charts representing Size (FP) versus Duration (Months) and Size (FP) versus Project Cost (Euros).

do not fail while large projects often do' might play a role here. An interesting side-effect of the observation is that nowadays software companies, in an attempt to become more agile, tend to opt for small releases (however this is not always the case: our sample holds two large Scrum releases of resp. 1,067 FPs and 4,600 FPs). Yet maybe active steering on economy of scale can be an equally effective – or even more effective – improvement strategy for software companies. We did not study this side-effect; however future research on the background of economy of scale versus agile development methods might

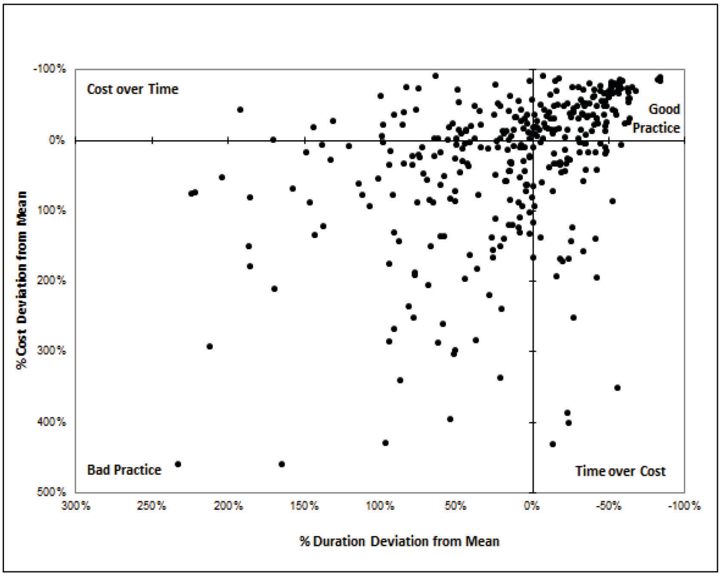


Figure 3.2: The cost duration matrix (see Subsection 2.3.2 for a detailed description of the model and its four quadrants).

help software companies to find an optimum on project size when building a portfolio.

3.4.2. Mapping on the Cost Duration Matrix

As a second step in the analysis all projects from the repository are classified in the four quadrants of a *cost duration matrix*, by combining two plotter charts:

1. A chart (top in Figure 3.1) where all projects from the repository (N = 352) are plotted in *project size* (FP) versus *project duration* (Months). This plotter chart indicates what projects score below the average trend line (M = 7.995, SD = 5.089, r^2 = 0.21) with regard to project duration, meaning the project duration is shorter than average, and what projects score above the average trend line, meaning the project duration is longer than average.
2. A chart (bottom in Figure 3.1) where all projects from the repository (N = 352) are plotted in *project size* (FP) versus *project cost* (Euros). This plotter chart indicates what projects score below the average trend line

Table 3.3. Average performance per cost duration quadrant.

Performance Indicator	Good Practice	CoT	ToC	Bad Practice
Cost per FP (EUR/FP)	1,285	1,448	3,834	5,285
FP per Hour	0.082	0.064	0.028	0.021
Days per FP	0.64	0.92	0.75	1.77
Defects per FP	0.06	0.20	0.19	0.27
Number in Sample	114	55	51	131

(M = 750777, SD = 1019949, $r^2 = 0.56$) with regard to project cost, meaning the project cost are less than average, and what projects score above the average trend line, meaning the project cost are higher than average.

For each project the measure of deviation from the average trend line is calculated and expressed in a percentage; negative when below the average trend line, positive when above the trend line. Based on this percentage all projects from the repository are plotted in a matrix, resulting in four quadrants. Each quadrant is characterized by the measure of negative or positive deviation from the average trend (see Figure 3.2).

Table 3.3 gives an inventory of the most important performance indicators with regard to the four *cost duration quadrants*. The table clearly indicates that projects that score as *good practice* on average show the best productivity, time-to-market, and process quality of all four quadrants. And for *bad practices* these performance indicators are the lowest of all four quadrants. The cost of a FP is for example for a *bad practice* as much as four times higher than the average cost for a FP of a *good practice*.

3.4.3. Analysis of Project Keywords

Once all projects have been classified in the *Cost Duration Matrix*, we analyze how the 56 project keywords are related to the four quadrants of this matrix (respective *good practice*, *bad practice*, *cost over time*, and *time over cost*). Two different perspectives are analyzed: the distribution over the four quadrants per number of projects and the distribution over the four quadrants based on project cost. In order to find significant differences between the two perspectives (i.e. number of projects and project cost) we use the chi-square test.

Table 3.4. Overview of factors strongly related to Good Practice.

Project Factor	% Good Practice	*N*	*p*
PPL Visual Basic	83	6	0.03
PPL FOCUS	80	5	0.06
KW Steady heartbeat	71	49	0.00
KW Fixed, experienced team	66	62	0.00
BD Data Warehouse & BI	61	18	0.02
PPL PowerBuilder	60	5	0.14
DM Agile (Scrum)	56	45	0.00
PPL SQL	56	9	0.10
BD Organization	52	31	0.02
KW Release-based (one application)	50	144	0.00

After removal of non-significant results we base the interpretation at the percentage number of projects per *cost duration quadrant*; resulting in an average percentage per quadrant for all 56 research aspects. A summary of the total analysis, including the result of the chi-square test, is established in the appendix of the accompanying technical report (Huijgens et al., 2013b).

3.4.4. Success Factors and Failure Factors

As a fourth and last step we identify specific success factors and failure factors for the composition of software engineering project portfolios, by analyzing how the scores of the different research aspects relate to the *cost duration quadrants*. We identify what aspects are strongly related (50% or more) to a high percentage of *good practice* and what aspects are strongly related (50% or more) to a high percentage of *bad practice*.

3.5. Evaluation

In this section we evaluate results and implications of the study. Analysis results in an inventory of factors that are strongly related to a high percentage of *good practice* (also referred at as success factors) and factors that are strongly related to a high percentage of *bad practice* (failure factors). To create better insight in the measure of modification of the factors, both success and failure factors are classified into categories of factors that are IT-organizational, business-organizational, and factors that are primarily technical.

In total 10 research aspects were found to be strongly related to *good practice* (success factors), and 13 research aspects strongly related to *bad practice* (failure factors). Besides that we found 1 factor that is strongly related to a high percentage of *cost over time*, and 2 factors that could be related to a high percentage of *time over cost*.

3.5.1. Factors Excluded from the Inventory

In order to find significant differences between the two perspectives (i.e. number of projects and project cost) we use the chi-square test. We have found a number of significant differences between the two perspectives: 8 out of 56 factors are excluded from the interpretation. One factor, KW Package off-the-shelf (without customization), scores as related to a high percentage of *good practice*; however only one such project was in the sample. With regard to 6 factors we found a low significance between the percentages *good practice*, *cost over time*, *time over cost*, and *bad practice*, measured on number of projects and the percentages measured on project cost (χ^2 of lower than 5). These 6 excluded factors are:

- Primary programming language *Oracle*, $\chi^2(1, N = 29) = 0.38, p < .01$.
- Business domain *Finance & Risk*, $\chi^2(1, N = 54) = 3.27, p < .01$.
- Primary programming language *RPG*, $\chi^2(1, N = 6) = 2.68, p = .18$.
- Keyword *Phased project*, $\chi^2(1, N = 65) = 4.26, p = .05$.
- Development class *Minor Enhancement*, $\chi^2(1, N = 27) = 2.78, p = .04$.
- Primary programming language *3GL*, $\chi^2(1, N = 8) = 3.18, p = .19$.

Two of these factors, Primary programming language *ORACLE* (86% *good practice* based on number of projects) and Business domain *Finance & Risk* (69% *good practice* based on number of projects) both score as factors that are strongly related to a high percentage of *good practice*, for which a remark is in place. Primary programming language *ORACLE* scores with a relation to a high percentage of *good practice* when looked upon from number of projects (86%), however it also has a high percentage of *bad practice* when looked upon from project cost (73%).

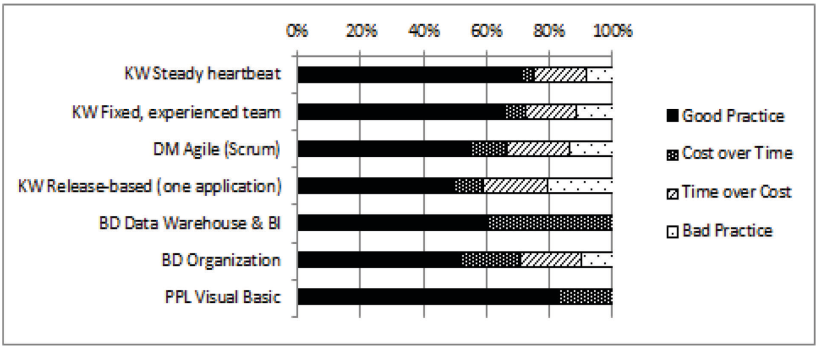


Figure 3.3: Overview of success factors.

This effect is caused by the fact that a sample of 29 primary programming language *ORACLE* projects is analyzed, including 26 very well performing small releases belonging to two applications (Huijgens & van Solingen, 2013a). The other three medium sized Primary programming language *ORACLE* projects did not perform well; two score as a *bad practice*, and one scores in the *time over cost* quadrant, resulting in a low χ^2-score. This also affects the score for Business domain *Finance & Risk*, as all 26 good performing projects are performed within this domain, including one medium sized badly performing project that influences the score with regard to project cost, again leading to a low χ^2-value.

Finally, a factor that is excluded from the inventory is the category Primary programming language *Other*: representing a collection of 9 primary programming languages that were recorded for only one project each in the repository.

3.5.2. Factors Strongly Related to Good Practice

We identified 10 factors that were strongly related to a high percentage of *good practice*. After testing for statistical significance we found that of those 10 factors 7 satisfied the alternative hypothesis (H_1) (see Figure 3.3).

Four factors of these 7 are tested to be strongly significant ($p < 0.01$) to influencing project success (high probability to end up as a *good practice*), and due to that can be specified as strongly significant success factors for software projects (ranking based on probability of success):

1. Steady heartbeat;

2. Fixed, experienced team;
3. Agile (Scrum) ;
4. Release-based (one application) .

It is likely that these four strongly significant success factors are related to each other, since release-based working, a steady heartbeat, and a fixed, experienced team, are in fact preconditions of an agile (Scrum) way of working. However keep in mind that not all projects where factor 1, 2, or 4 was applicable used an agile delivery model. In fact all types of projects can adopt these success factors, without opting for a specific delivery model. The promising idea behind these four success factors is that they all can be implemented relatively easy and fast in any software engineering company: no big organizational or technical changes are needed to start working according to these success factors. In a way an organization can decide to start working this way by tomorrow.

Besides these four strongly significant success factors, we found out of the group of 7 factors that were strongly related to *good practice*, 3 factors with a significant probability ($p < 0.05$) to perform as a *good practice* (ranking based on probability):

5. Business domain Data Warehouse & BI;
6. Business domain Organization;
7. Primary programming language Visual Basic.

One could argue that a difference is applicable with regard to the three success factors above with the success factors 1 to 4 on the fact that (where 1 to 4 seems relatively easy to implement) 5 to 7 are more difficult to change. A software company cannot simply change its business organization structure overnight, and opting for another programming language asks for a more long term approach.

As an example we do see implementations in practice where a company actively builds its application portfolio on a limited number of programming languages, however in many cases the reason behind such an improvement strategy lies in steering on shortage versus availability of language-specific developers (avoiding of high labor cost) instead of implementing continuous performance improvement.

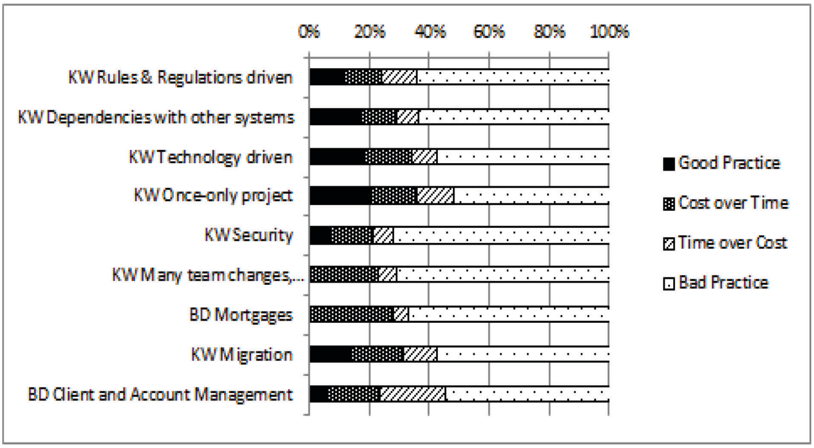


Figure 3.4: Overview of Failure Factors.

Although we found a strong significance for primary programming language *Visual Basic* a remark is in place: 6 projects were analyzed of which 5 scored as *good practice* and one as *cost over time*. The projects took place in two different companies and varied in size from 27 to 586 FPs.

3.5.3. Factors Strongly Related to Bad Practice

We identified 13 factors that were strongly related to a high percentage of *bad practice*. After testing for statistical significance we found that of those 13 factors 9 satisfied the alternative hypothesis (H_1) (see Figure 3.4). 4 Factors of these 9 are tested to be strongly significant (p < 0.01) to influencing project failure (high probability to end up as a *bad practice*), and due to that can be specified as strongly significant failure factors for software projects (ranking based on probability of failure):

1. Rules & regulations driven;
2. Dependencies with other systems;
3. Technology driven;
4. Once-only project.

An interesting relation can be assumed here between the strongly significant success factor *Release-based*, and the strongly significant failure factor *Once-only project*. Although our research does not show, we assume that starting up a once-only project, including having to do things for the first time, and for one project only, leads to a high probability of ending in between *bad practice*. On the other hand the repeating character of release-based working, including the effect of learning on-the-job and creating an experienced team, creates a high probability to end up as a *good practice*.

Next to these four strongly significant failure factors, we found that 5 factors from the group of strongly related to a high percentage of *bad practice* showed a significant ($p < 0.05$) probability to perform as a *bad practice* (ranking based on probability of failure):

5. Security;
6. Many team changes, inexperienced team;
7. Business domain Mortgages;
8. Migration;
9. Business domain Client & Account Management.

3.5.4. Factors Related to CoT and ToC

We did not find any factors that were strongly related to a high percentage of *cost over time* (cost lower than average, duration longer than average) or *time over cost* (cost higher than average, duration shorter than average) that where significant ($p < 0.05$) for respectively *cost over time* or *time over cost*.

3.6. Discussion

In this section we discuss the most important limitations with regard to the study.

3.6.1. Research on an Existing Repository

Owing to the fact that the data of software engineering projects that is used for the research was collected in a practical setting and primary for practical purposes (e.g. collect historic project data as a source for estimation and benchmarking), this study must emphatically been seen as the result of analysis that was performed on an existing measurement repository. As a consequence of that we had to cope with the data that was available: no additional data was to be collected at a later stage. The dataset only contained data of finalized (successful) projects; no data of prematurely stopped or failed projects was available. In particular for the identification of bad practices, the study of failing projects will be relevant.

3.6.2. Uncertainties related to Software Metrics

A remark is in place with regard to uncertainty ranges and data collection problems that exist in repositories of software metrics data. Variation ranges of about ±15% between projects and organizations due to the counting rules for data (Boehm & Sullivan, 2000b). Data collection problems such as missing values, cost segmentation due to different counting methods, too many factor levels, and not enough factors (Angelis, Stamelos, & Morisio, 2001).

Algorithmic estimation models based on historic project data (e.g. SLIM, Function Points, COCOMO 2) do not model the factors affecting productivity very well (Kemerer, 1987). All are examples of uncertainties that might apply to our research data too, in spite of a great effort that was put in quality assurance activities during data collection by the teams that helped building our research repository.

However, a majority of the mentioned problems are related to estimation based on algorithmic models derived from historic data, while this is not the case in our research. We divided our research dataset in relatively large parts (the four *cost duration quadrants*) and applied impact factors on two of these. The uncertainties that do apply on algorithmic-based effort and schedule estimation are not automatically applicable to our research too.

Looking at the research challenge raised by (Boehm & Sullivan, 2000b) that "we need to learn how to think about and manage software development as an investment activity, the goal of which is to create maximum value for the resources invested" we feel that (unless any data collection problems and uncertainties) the goal of our study to help enterprises build *good practice* project portfolios is a step ahead towards solving that challenge.

3.6.3. Business Domain and Programming Language

Due to the upfront collection of project data, we could not exactly identify the cause behind the fact that programming language *Visual Basic* and the business domains *Data Warehouse & BI* and *Organization* turn out to be a success factor, and that on the other hand business domains *Client & Account Management* and *Mortgages* ended up to be a *bad practice*.

One might assume that in the three given companies the best people dealt with *Data Warehouse & BI* and *Organization* projects, and that *Visual Basic* skills were stronger than other skills. And on the opposite, that within the business domains *Client & Account Management* and *Mortgages* had to deal with lower skilled resources.

However, it is more likely that *Data Warehouse & BI*, *Organization* and *Visual Basic* project environments on average are less complex than others, and due to that end up in the *good practice* quadrant more often. And that *Client & Account Management* and *Mortgages* related project environments on average are more complex than others (e.g. legacy, migration, and dependencies with other departments or domains play a relatively large role here). Benchmarks, e.g. (ISBSG, 2014), do confirm this assumption.

3.6.4. Generalization

Analysis of the performance of the separate companies showed that *Company* itself was not a distinguishing factor that could be strongly related to either a high percentage of *good practice* or *bad practice*. We assume that the results of our research generalize given this finding (depending on the extent to which the three companies are representative), in combination with the fact that our research included three different software companies.

3.7. Related Work

In the following, we discuss the contribution of our study in the context of earlier research. Much has been written on identification of success and failure factors for software engineering, in many cases with reference to software process improvement (SPI), a popular process-based approach to delivering improvements in software products from the 80s (Hall, Rainer, & Baddoo, 2002).

(Reel, 1999) argues that in software, more “advanced” technologies are far less critical to improving practice than five essential factors to managing a successful software project: start on the right foot, maintain momentum, track progress, make smart decisions, and institutionalize post-mortem analyses.

Dybå (2005) did find strong support for SPI success to be positively associated with business orientation, employee participation, concern for measurement, and exploitation of existing knowledge, and partial support for

SPI success to be positively associated with involved leadership and exploration of new knowledge. In another study Dybå (2003) showed that small organizations reported that they implement SPI elements as effectively as large organizations, and in turn, achieve high organizational performance.

(Niazi et al., 2006) identified seven success factors for SPI: management support, training, awareness, allocation of resources, staff involvement, experienced staff and defined SPI implementation methodology. Rainer and Hall (2002) found four SPI success factors: reviews, standards and procedures, training and mentoring, and experienced staff.

In the 90s more advanced process improvement models such as CMMI, and ISO's SPICE were introduced (Niazi et al., 2003), leading to supplementary research on success and failure factors. (Stelzer & Mellis, 1998) describe ten process oriented factors that affect organizational change in software process improvement initiatives based on the CMM and ISO quality standards.

(Niazi et al., 2003) inventoried three categories of critical success factors: awareness, organizational, and support. A number of critical barriers were identified (a.o. lack of resources, time pressure, inexperienced staff, organizational politics, and lack of formal methodology).

(Procaccino et al., 2002) found the most important factors for project success to be the presence of a committed sponsor and the level of confidence that the customers and users have in the project manager and development team. (Charette, 2005) names twelve factors why software fails so often, however there is no clear link with the results from our research with regard to these factors. The common idea in these papers is that success and failure were interconnected with process-based activities: in other words, follow the process and success will come. Looking at the results of our research, a close link with process related research, often based on software projects performed during the last two decennia of the former century, seems absent due to the fact that we did not study this: no clearly process related (in terms of SPI, CMMI, or ISO) factors were present in our list of project factors.

Recent research focusing on the factors that affect success and failure of agile software engineering is more similar. This seems relevant, since more than 80% of organizations now following an agile approach (Fitzgerald, et al., 2013). (Solingen & Berghout, 1999) defined a stepwise approach to derive metrics from a software company's goal. (Chow et al., 2008) state that, 'as long as an agile project picks a high-caliber team, practices rigorous agile software engineering techniques and executes a correct agile-style delivery strategy; the project could be likely to be successful'. Three other factors that could be critical to certain success dimensions are a strict agile project management process, an agile-friendly team environment, and a strong customer involvement. Although not stated in the same words a similarity with the results from our research is noticeable here. On the other hand we did not find evidence (possibly due to the fact that we did not examine this subject in the scope of this study) that some assumed prerequisites for success of agile projects such as strong executive support, strong sponsor commitment, ready availability of physical agile facility, or agile-appropriate project types, are actually critical factors for success.

(Misra et al., 2009) found nine factors that have statistically significant relationship with success in adopting agile software development practices: customer satisfaction, customer collaboration, customer commitment, decision time, corporate culture, control, personal characteristics, societal culture, and training and learning. No clear link with results from our research is found here.

(Sutherland et al., 2007) studied best practices in a globally distributed Scrum environment, with fixed, experienced agile teams, and comes up with, as the paper states 'the most productive Java projects ever documented'. Maybe stated differently, however, a match with our findings is obvious here. Otherwise, the idea that agile is always linked to project success is not shared by (Estler et al., 2012) stating that 'choosing an agile rather than a structured process does not appear to be a crucial decision for globally distributed projects.'

Our research indicates a relation between agile (Scrum) and economy of scale (project size). (Boehm, 2006a) states that 'the most widely adopted agile method has been XP, whose major technical premise was that its combination of customer collocation, short development increments, simple design, pair programming, refactoring, and continuous integration would flatten the cost-of change-vs.-time curve. However, data reported so far indicate that this flattening does not take place for larger projects.' Apparently the success of agile does not hold for larger sized projects, something that matches an assumption in our research, since no medium large or large projects were performed based at an agile delivery method. The connection between agile (Scrum) and project size is not clear yet and seems a challenge for future research.

A fascinating gap in related research turns up with relation to the finding from our research that a programming language (*Visual Basic*) and specific business domains (*Data Warehouse & BI*, *Organization*) are significant for project success. It might be surprising that a view at improvement, focusing at benchmarking the outcome of the software process, instead on the software process itself, seems less represented in related work. (Jones, 1995) inventories successful and unsuccessful project technologies, stating that unsuccessful projects are related to a lack of measurement activities, and conversely, successful projects are related to accurate measurement and analysis activities. (Jones, 2000) inventoried a large amount of business domain and programming language specific effects on project performance. In the 90s he performed extensive research on performance aspects related to domain-specific and language-specific software engineering, however since that not many additional research on this area, and especially on the effects of agile delivery models, seem to be performed.

Regarding our finding that rules & regulations driven projects are significant for failure, such as high costs, long durations and low quality, Gong and (Janssen, 2012) defined principles for creating flexibility and agility when implementing new or revised policies into business processes. Besides that, approaches such as lean government (Janssen & Estevez, 2013) and include risk management in enterprise architectures (Janssen & Klievink, 2010) might help to reduce risks with regard to rules & regulations driven projects.

(Premrai et al., 2005) investigated how software project productivity had changed over time, finding that an improving trend was measured, however less marked since 1990. The trend varied over companies and business sectors, a finding that matches the result of our research with regard to differentiation over business domains.

3.8. Conclusions and Future Work

We found 7 success factors for software projects: (1) Steady heartbeat, (2) Fixed, experienced team, (3) Agile (Scrum), (4) Release-based (one application), (5) Business domain Data Warehouse & BI, (6) Business domain Organization, and (7) Primary programming language Visual Basic.

We found 9 failure factors for software projects: (1) Rules & regulations driven, (2) Dependencies with other systems, (3) Technology driven, (4) Once-only project, (5) Security, (6) Many team changes, inexperienced team, (7) Business domain Mortgages, (8) Business domain Client & Account Management, and (9) Migration. Based on the findings we identify the following guidelines for practice when aiming for a *good practice* project portfolio:

1. Avoid *bad practice* by steering on limitation of inter-dependencies between projects and systems, stay away from unnecessary team changes, and build teams where possible with experienced team-members.
2. Create *good practice* by actively steering on organizing software development in a release-based way, set up fixed teams with experienced team-members, implement a steady heartbeat, and go for an agile (Scrum) delivery approach.
3. When setting up a once-only project within a software project portfolio, pay special attention to standardization and limitation of procedures to smoothen the projects progress, re-use of knowledge of other once-only projects (Lessons Learned), and implementation of a learning cycle.
4. Implement a long- and medium term portfolio strategy, including a learning capability, to avoid *bad practice* by limitation (where possible) of projects that are characterized as technology driven, rules & regulations driven, migration of data, and security issues.

3.8.1. Future Work

An interesting side-effect of the observation is that nowadays software companies, in an attempt to become more agile, tend to opt for small releases. Yet maybe active steering on economy of scale can be an equally effective – or even more effective – improvement strategy for software companies. We did not study this side-effect; however future research on the background of economy of scale versus agile development methods might help software companies to find an optimum on project size when building a portfolio. Another aspect that was not covered in our research, yet might help companies to achieve continuous improvement, is whether the software companies did really improve their performance over time. During the six year measurement period many improvement actions were undertaken, yet how successful were these actions in the end? Future research (on our dataset) might reveal this.

3.9. Acknowledgments

We thank Hans Jägers, professor emeritus from the University of Amsterdam, Rob de Munnik (QSM Europe), Bart Griffioen, Georgios Gousios, Steven Raemaekers, and all other reviewers for their valuable contributions.

3.10. Addendum

Inspired by new and sometimes improved insights into the statistical tests to be used for software analytics, we challenged the tests performed in the original paper that was published in 2014. For this purpose we performed Spearman Rank Correlation tests for both the original subset of 352 projects that were in scope of Chapter 3, and the latest version of the EBSPM research repository (Huijgens, 2017a) with 498 projects in scope. Because we performed a larger number of tests, we added multiple comparisons p-value adjustment, to prevent from p-values less than 0.05 purely by chance, even if all null hypotheses are really true. For this purpose, we applied the Bonferroni correction to control the familywise error rate. As a more powerful method for adjusting the false discovery rate we applied the Benjamini-Hochberg procedure.

Because the EBSPM research repository grew over the four-year period of our study, we also applied both the correlation test and the p-value adjustments on the total repository of 498 projects that we collected at the end of the study period.

3.10.1.Pairwise Correlation and P-value adjustment

Table 3.5 shows a summary of the results of the tests regarding success factors that are strongly significant with Good Practice, performed on the initial subset of 352 projects as used within the scope of Chapter 3. When compared to the original outcomes, as depicted in Figure 3.3, it shows that the outcome in terms of project factors that are strongly significant for Good Practice matches the original study. However, the ranking of the factors is slightly different, and the factors Business Domain Organization and Programming

Table 3.5. Overview of Pairwise Correlation testing, with multiple comparisons p-value adjustment related to Good Practice on the subset of 352 projects in scope of Chapter 3 (only significant factors with an adjusted (BH) p-value < 0.01 are shown).

Project Factor strongly significant for Good Practice	Raw p-value	*Adjusted Bonferroni*	*Adjusted Benjamini-Hochberg*
Steady Heartbeat	8.956e-11	2.05988e-09	2.05988e-09
Fixed, experienced team	3.935e-10	9.05050e-09	4.525250e-09
Release-based	8.180e-10	1.88140e-08	6.271333e-09
Agile (Scrum)	2.806e-05	6.45380e-04	1.290760e-04
Single application	1.002e-04	2.30460e-03	3.84100e-04
Business Domain Data Warehouse & BI	9.442e-04	2.17166e-02	3.102371e-03

Language Visual Basic are not in the results of the new tests, mainly due to the additional p-value adjustments.

The results of the tests performed over the whole repository of 498 projects are summarized in Table 3.6. Except for the ranking of factors, no major differences are to be found with the tests on the smaller initial subset of projects. The plot in Figure 3.5 illustrates the different p-value adjustments of the applied tests.

Table 3.7 summarizes the results of the tests regarding failure factors that are strongly significant with Bad Practice, performed on the initial subset of

Table 3.6. Overview of Pairwise Correlation testing, with multiple comparisons p-value adjustment related to Good Practice on all 489 projects in the EBSPM research repository (only significant factors with an adjusted (BH) p-value < 0.01 are shown).

Project Factor strongly significant for Good Practice	Raw p-value	*Adjusted Bonferroni*	*Adjusted Benjamini-Hochberg*
Release-based	6.352e-12	1.2704e-10	1.27040e-10
Steady Heartbeat	2.003e-08	4.0060e-07	2.00300e-07
Fixed, experienced team	6.270e-08	1.2540e-06	4.18000e-07
Single application	4.345e-06	8.6900e-05	2.17250e-05
Agile (Scrum)	1.281e-05	2.5620e-04	5.12400e-05
Business Domain Data Warehouse & BI	7.916e-05	1.82068e-03	3.034467e-04

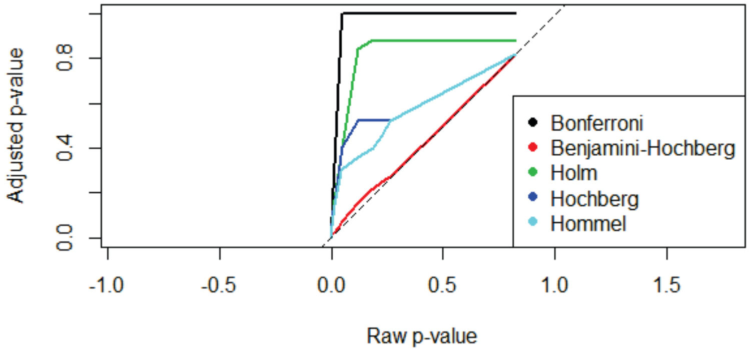


Figure 3.5: Plot of adjusted p-values after testing multiple comparisons related to Good Practice for all 489 projects in the EBSPM research repository.

352 projects as used within the scope of Chapter 3. When compared to the original outcomes, as depicted in Figure 3.4, it shows that again no major differences are found in the factors itself, only the ranking deviates at some points. This also applies to the results of the tests over the whole repository of 498 projects, as summarized in Table 3.8, although dependencies with other applications is not a significant failure factor in the repository as a whole

Table 3.7. Overview of Pairwise Correlation testing, with multiple comparisons p-value adjustment related to Bad Practice on the subset of 352 projects in scope of Chapter 3 (only significant factors with an adjusted (BH) p-value < 0.01 are shown).

Project Factor strongly significant for Bad Practice	Raw p-value	*Adjusted Bonferroni*	*Adjusted Benjamini-Hochberg*
Migration project	1.350e-04	3.37500e-03	6.154167e-04
Rules & Regulations driven project	1.350e-04	3.37500e-03	6.154167e-04
Once-only project	3.377e-04	3.69250e-03	6.154167e-04
Many team changes, unexperienced team	3.513e-04	8.44250e-03	1.097812e-03
Dependencies with other applications	5.609e-04	8.78250e-03	1.097812e-03
Phased project	2.495e-03	6.23750e-02	5.670455e-03
Technology-driven project	3.227e-03	8.06750e-02	6.722917e-03

Table 3.8. Overview of Pairwise Correlation testing, with multiple comparisons p-value adjustment related to Bad Practice on all 489 projects in the EBSPM research repository (only significant factors with an adjusted (BH) p-value < 0.01 are shown).

Project Factor strongly significant for Bad Practice	Raw p-value	*Adjusted Bonferroni*	*Adjusted Benjamini-Hochberg*
Rules & Regulations driven project	4.874e-06	1.16976e-04	2.339520e-05
Migration project	3.025e-05	7.26000e-04	1.210000e-04
Many team changes, unexperienced team	5.715e-05	1.37160e-03	1.959429e-04
Phased project	8.155e-05	1.79410e-03	2.30952e-04
Once-only project	8.850e-05	2.12400e-03	2.30952e-04
Technology-driven project	9.623e-05	2.30952e-03	2.30952e-04

anymore. Summarizing, the additional Pairwise Correlation tests that we performed – including the p-value adjustments – confirm the findings in the original 2014 study. Although differences occur in the ranking of success- and failure factors, the factors itself are the same in both the original study and in the additional tests. However, maybe linear regression testing would be a better option, because this will also consider the effects of factors on each other.

We found that a release process that performs above average on cost and duration, satisfies stakeholders through fast response and direct value, even when the reliability and availability of the actual system is weak.

Photo by Sven Scheuermeier

4. The Cecil-Case: Managing Legacy Evolution

C*ontext:* In this chapter, we attempt to understand what contributes to a successful process for managing legacy system evolution. *Objectives:* We provide an analysis of a number of key performance indicators such as cost, duration, and defects. By normalizing through function points, we furthermore compare to a larger benchmark. *Method:* To do so we performed a mixed, retrospective case study on a series of nine software releases and eight single once-only releases, all performing on a single, legacy software system, in a West-European telecom company. We interviewed eleven stakeholders that were closely involved in the subject software releases. *Results:* As a result, we listed a number of observations from the quantitative and qualitative analysis. *Conclusions:* We found that a release process that performs above average on cost and duration satisfies stakeholders through fast response and direct value, even when the reliability and availability of the actual system is weak.

This chapter was published as *Success factors in managing legacy system evolution: a case study* in the proceedings of the 38th International Conference on Software and Systems Process (ICSSP 2016) (Huijgens, van Deursen, van Solingen, 2016d).

4.1. Introduction

Managing legacy systems, and especially linking the building of new software with evolution of legacy systems is a big challenge for many companies (Boehm, 2006b) (Deursen et al., 1999). For this study we analyzed a series of nine software releases (the *Cecil releases*), performed in a Belgian telecom company (in the remaining of this chapter indicated as *BelTel*), that is characterized by highly satisfied stakeholders. This study aims at analyzing software releases to only one system, that were conducted in different ways. The *Cecil releases* are typically built from quick wins; fast and small enhancements on a system by a single dedicated Scrum team. All releases were performed on one Customer Relationship Management (CRM) system - named the Divine system, which is a five-year-old legacy system that is planned to be replaced because of ongoing reliability and availability problems.

While performing our case study, four things puzzled us. First, stakeholders were largely satisfied with the deliveries of the Cecil team. Second, the Cecil releases were assessed all to be ‘best-in-class’ in terms of cost, duration, and defects found, when benchmarked against other software deliveries in our research repository. Third, besides the Cecil releases another eight releases were performed on the Divine legacy system, outside of the scope of the Cecil team. And these were all assessed as not being ‘best-in-class’. And finally, the Divine system itself was performing badly in terms of reliability and availability.

The goal of this chapter is to understand what contributes to a successful process for managing legacy system evolution. To reach this goal, we provide an analysis of a number of key performance indicators such as cost, duration, and defects. By normalizing through function points, we furthermore compare to a larger benchmark. Furthermore, to understand in depth what contributed to the success of the Cecil releases, we conduct in depth interviews with eleven people close involved with Cecil.

The remainder of this chapter is organized in the following way: In Section 4.2 we outline the experimental setup for our case study. In Section 4.3 the research approach that we apply is described. Sections 4.4 and 4.5 are about the results of our study. In Section 4.6 we discuss the results, compare them with state of the art and we discuss threats to validity. In Section 4.7 we discuss related work. Finally, Section 4.8 includes conclusions.

4.2. Experimental Setup

4.2.1. Context

We analyzed the Cecil releases and non-Cecil releases, performed over a period of one year on a single software system in *BelTel*, a Belgian telecom company. We performed both quantitative and qualitative analysis, the latter by performing open-ended, non-structured interviews with stakeholders on the backgrounds of the success of the releases. Regarding confidentiality of data, the names of companies, systems, releases, and people are made anonymous in this study. To improve the readability of this study, we provide definitions of four used acronyms:

Cecil releases

A series of nine software releases, Cecil releases, are performed release-based, with a fixed team of six persons, with a steady heartbeat (Go Live every six weeks), and a Scrum approach. Within the Cecil releases only small enhancements are included; also in former years identified as CRM Quick Wins. These quick wins are primarily GUI-driven and meant to solve process issues in the Divine system that's mainly used by agents (front-office employees of *BelTel* that have contact with customers through various channels (e.g. telephone, call centers, email, and chat). Driven by an attempt to speed up the software delivery process, in 2014 a decision was made to setup a fixed team that was budgeted only once each year. This means that a budget was approved for capacity of the team for a whole year.

The Cecil release team consists of six people that all work fulltime for the team. From *BelTel* itself these are a product owner and a business analyst, from *IndSup-A*, *BelTel*'s main Indian supplier, three software developers, and one tester from the main supplier that is responsible for user acceptance and regression testing. Besides that, an enterprise architect is involved in design activities on an ad hoc basis, and a release manager performs the integration once ready for release.

Non-Cecil releases

Eight software releases on the Divine system that were performed once-only. Contrary to the Cecil releases these releases are characterized by a new team setup for every release, in advance governance and budget approval for each release, plan-driven approach.

The eight non-Cecil releases were performed as once-only releases, meaning that a team was setup preceding every single release and closed down once the release was finalized. Only few people within *BelTel* were still to be found that joined in a non-Cecil release; of the interview participants mentioned in Table 4.4 only P1, P5, P8, P9, and P11 were involved in any way in these releases.

Divine system

The Customer Relationship Management (CRM) software system on which both the Cecil releases and non-Cecil releases are performed. A complicating factor in this study is the fact that the Divine system is a legacy system (planned to be replaced) that faces severe reliability and availability issues.

BelTel

The Belgian telecom company where Cecil releases and non-Cecil releases are performed. A repository with data of approximately 95 finalized software projects that is collected over time in *BelTel* is used in this study as a reference for benchmarking purposes.

Within *BelTel* a company standard allows for eight pre-planned production releases per year; each includes a number of projects and releases (from Cecil, Divine, or other teams), that jointly move on to user acceptance testing and integration into the production environment.

Following conventions in use at *BelTel*, we use the term release with two different meanings. Release is used to indicate a specific software project that is performed in a release-based way. In this chapter we use the term release for those cases. Besides that, the term release is used to indicate a combined set of projects and releases for integration into the production environment. In this chapter we use the term *production release* if this is the case. Within the *BelTel* practice, these *production releases* are deployed into the production environment in yearly eight subsequent production releases; this applies for both Cecil releases and non-Cecil releases too.

4.3. Research Questions

Based on this we defined the following research questions:

RQ4.1: To what extent can a release-based iterative process be successfully used to manage the evolution of a legacy system?

RQ4.2: To what extent play known factors a role in this success?

RQ4.3: What specific factors contributed to this successful way of managing legacy system evolution?

The case study that we performed is a mixed study, it includes both quantitative and qualitative research on the subject releases (Runeson et al. 2012) (Yin, 2008). The analysis falls apart in two parts. First, we quantitatively analyzed the Cecil and non-Cecil release data that we collected on a series of releases over time and compare the outcomes with earlier research on best-in-class software releases (Huijgens & van Solingen, 2013a). Second, we conducted qualitative research by performing open-ended, non-structured interviews with members of the release-teams, and internal customers of *BelTel*, that made use of the deliverables of the Cecil team and the non-Cecil teams.

4.3.1. Data Collection Procedure

For our research we used two types of data: data that was collected in the period before we started our study, as part of the operational measurement practice within *BelTel*, and data we collected specifically for our research. The first consists of artifacts collected over time on these software releases, supplemented with data from two releases that finalized in the two previous years. Among others the following artifacts were available for our research:

- Quantitative data that was recorded in a measurement repository on both the Cecil releases and the non-Cecil releases, holding measurements such as size, cost, duration, number of defects, and planning data on cost and duration.
- Performance dashboards and *Project Close Reports* on the finalized Cecil releases and non-Cecil releases.
- Logged data in the tool that was used for backlog management (Scrumwise[1]), including User Stories, Story Points, an activity log, attached documents, and comments on backlog items.

[1] https://www.scrumwise.com

- Technical design documents of the Cecil releases and non-Cecil releases, usually prepared by members of the Cecil team or members of the once-only non-Cecil releases teams

Besides collecting existing data within *BelTel* we performed interviews with key stakeholders within *BelTel* that were involved in the Cecil releases. The stakeholders were all involved in the operational practice of the Cecil releases; the list of stakeholders is setup in close cooperation with the business analyst and with the product owner. See Table 4.4 at page 81 for an overview of interviewed stakeholders. All interviews were performed on a one-to-one basis between the author of this thesis [interviewer] and one specific stakeholder [interviewee], except for two interviews where two interview participants were combined in one interview at request of the participants. The interviews were based on the following questions, where applicable sub questions are asked to reveal backgrounds or to clarify misunderstandings or indistinctness:

1. Can you give some backgrounds on your role in the Cecil releases and non-Cecil releases?
2. What top-5 aspects did influence the releases in a positive way?
3. What top-5 issues need improvement?
4. In what way did the series of Cecil releases evolve over time (e.g. process changes, changes in way of working, team changes, changes in roles)?
5. In what way did the Divine system evolve over time (e.g. new functionality, enhancements, lifecycle of the system)?
6. Are there any other important things to mention?

4.3.2. Quantitative Analysis

In order to perform quantitative analysis on the collected data of Cecil- and non-Cecil releases, we calculated three performance indicators and compared the outcomes with those of our earlier published research paper on best-in-class software releases (Huijgens & van Solingen, 2013a). We calculated the following performance indicators:

1. *Cost per FP*: a weighted average of the summarized project cost in Euros divided by the summarized project size in FPs (weighting factor is project size).

2. *Duration per FP*: a weighted average of the summarized project duration in calendar days divided by the summarized project size in FPs (weighting factor is project size).
3. *Defects per FP*: a weighted average of the summarized number of defects found during each release divided by the summarized project size in FPs (weighting factor is project size).

In order to quantify the measure of success and failure we used a *Cost Duration Matrix* (see Subsection 2.3.2), a model developed in earlier research (Huijgens et al., 2014c) that compares the performance of finalized software projects in terms of cost, duration, and number of defects found during development. We compared the performance and characterizations of the Cecil releases with a series of 26 best-in-class software releases that were performed in another company and that we described in earlier research (Huijgens & van Solingen, 2013a). We did not incorporate this study as a separate chapter in this thesis, because the outcomes were confirmed and extended in a follow-up study that we performed and that is incorporated in Chapter 3 of this thesis.

4.3.3. Qualitative Analysis

All interviews are recorded digitally and transcribed to text files. The text files are analyzed by following a number of steps. First, we read the transcripts, and made notes about first impressions. Subsequently, we coded relevant pieces of the transcripts, by labelling relevant words, phrases, sentences, or sections. When applicable, we decided that something is relevant for us because the interview participant explicitly states the importance, because it surprises us, or because it is repeated in several places. Our aim was to look for conceptualization and underlying patterns. Then we decided which codes are the most important, and created categories by combining codes to logical themes and drop less important ones. Categories were labeled, and we decided which are the most relevant and how they are interconnected. We described the connections between them. Finally, we discussed the impact and implications of our observations, based on the results of the quantitative and qualitative analysis. We used triangulating evidence from multiple sources to obtain our final findings.

Table 4.1. Overview of the collected metrics for the CECIL releases.

	Go Live Date	Release Size (FPs)	Story Points	#User Stories	Release Cost (Euros)	Release Duration (Months)	# Defects (pre-release)
CECIL 2014 R3	4-2014	111	na	na	56,345	3.12	11
CECIL 2014 R4	5-2014	37	na	na	45,769	2.76	5
CECIL 2014 R6	8-2014	64	31 (0)	16 (0)	30,367	4.60	2
CECIL 2014 R7	5-2014	80	64 (0)	17 (0)	73,134	5.49	8
CECIL 2014 R8	11-2014	40	37 (0)	6 (0)	58,154	7.56	7
CECIL 2015 R1	1-2015	46	62 (0)	12 (0)	84,464	6.11	19
CECIL 2015 R2	2-2015	109	87 (0)	18 (0)	62,768	7.43	7
CECIL 2015 R3	4-2015	128	73 (16)	23 (4)	62,964	8.81	16
CECIL 2015 R4	5-2015	63	44 (0)	8 (0)	110,000	6.11	10

4.4. Quantitative Results

Between April 2014 and May 2015 nine Cecil releases were performed (see Table 4.1). Minimum release size was 37 FPs, maximum release size 128 FPs, medium size was 64 FPs. Cost of the releases varied from 30K Euro to 110K Euro, with a median of 63K Euro. Release duration took 2.76 to 8.81 months, with a median of 6.11 months. Eight non-Cecil releases (see Table 4.2) were

Table 4.2. Overview of the collected metrics for the non-CECIL releases.

	Go Live Date	Release Size (FPs)	Story Points	#User Stories	Release Cost (Euros)	Release Duration (Months)	# Defects (pre-release)
Divine Quick Wins 2012	11-2012	81	na	na	157,763	10.35	na
Divine Sales Orders	7-2013	110	na	na	358,883	11.60	18
Divine Detailed Rep.	10-2013	80	na	na	78,954	6.11	4
Divine Archival	12-2013	100	na	na	364,549	6.97	18
Divine Quick Wins 2013	12-2013	32	na	na	47,880	6.18	6
Divine Security	2-2014	61	na	na	181,483	6.93	5
Divine SO Tracking Tool	7-2014	22	na	na	69,761	9.20	7
Divine Stability Impr.	8-2014	8	na	na	22,930	8.31	na
Divine Quick Wins 2012	11-2012	81	na	na	157,763	10.35	na

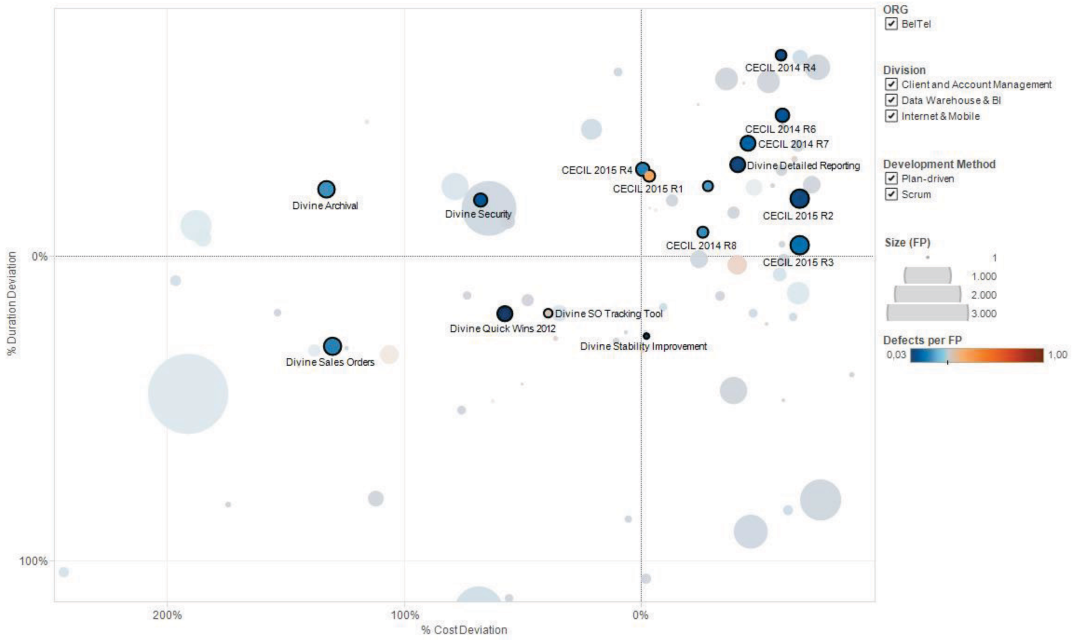


Figure. 4.1. Cost duration matrix of 95 finalized BELTEL releases with the CECIL releases and non-CECIL releases mapped at it.

deployed. Minimum release size was 8 FPs; maximum size was 110 FPs. Cost varied from 23 to 365 K Euro. Duration took between 6.11 and 11.6 months.

Table 4.3 gives an overview of a comparison between quantitative data of collected within *BelTel* on the finalized Cecil releases and 26 best-in-class software releases as analyzed in an earlier published research paper (Huijgens & van Solingen, 2013a).

In order to benchmark the performance of Cecil releases and non-Cecil releases in terms of cost, duration and number of defects against our research repository holding 95 software releases that were performed within *BelTel*, we used a model that we developed in earlier research; the *Cost Duration Matrix* (Huijgens et al., 2014c). The model can be used to compare a portfolio of releases to the benchmark, by means of a *cost duration matrix*, as shown in Figure. 4.1 for the 17 releases in scope of this chapter. The EBSPM Performance Dashboard, including the *cost duration matrix* is described in detail in Subsection 2.5.

Overall, the quantitative analysis tells us that all Cecil releases fall in the *good practice* category when mapped on a *cost duration matrix*, which means that all releases score better on *Cost per FP* and *Duration per FP* than the average of *BelTel* projects. When compared with the 26 software releases that were performed within another company (see Table 4.3), we observe that Cecil releases are on average approximately two-and-a-half times larger in size (FPs) than the best-in-class releases from earlier research (Huijgens & van Solingen, 2013a). Because of this and thanks to economies of scale the *Cost per FP* are approximately 10% to 20% lower. *Duration per FP* on the contrary is comparable to the average score of the best-in-class releases from (Huijgens & van Solingen, 2013a); average durations are longer, but the larger average size in FPs compensates this.

Finally, the comparison shows that the number of *Defects per FP* of both Cecil releases and non-Cecil releases is higher than that of the best-in-class releases in (Huijgens & van Solingen, 2013a), indicating a lower process quality. However, when benchmarked against our research repository Defects per FP of all but one of both Cecil releases and non-Cecil releases is better than the average score.

Observation 1: The performance of Cecil releases is, in terms of Duration per FP equal to, and in terms of Cost per FP 10 to 20% better than that of the earlier described best-in-class releases.

When Cecil releases are compared with the performance of non-Cecil releases within *BelTel* that are performed on the same Divine system, it shows that Cecil releases overall performed significantly better on both *Duration per FP* and *Cost per FP*. As possible explanations for this we assume that the additional startup time and cost needed for once-only releases, the learning effort and knowledge gap of non-Cecil once-only release teams, the extra time and cost for the proposal phase of a one-only release, and the overhead of once-only releases over a long-term, fixed, and experienced Cecil team play an important role. Besides that, we assume due to the type of requirements that a number of the non-Cecil releases are more nonfunctional than Cecil releases, leading to relatively less function points (non-functional requirements are not in scope of function point analysis).

Table 4.3. Key Performance Indicator Comparison as earlier published in (Huijgens & van Solingen, 2013a).

	CECIL Releases	non-CECIL releases	Best-in-Class Releases System A	Best-in-Class Releases System B
Number of Releases	9	8	13	13
Throughput (FPs)	678	494	415	349
Average Project Size (FPs)	75	62	32	27
Average Project Cost (Euros)	64,885	160,275	35,563	35,571
Average Project Duration (Months)	5.78	8.21	2.49	2.49
Average Number of Defects	9.44	9.67	na	1.23
Cost per FP (Euros)	861	2,595	1,114	1,325
Duration per FP (calendar days)	2.33	4.04	2.37	2.82
Defects per FP	0.13	0.14	na	0.05

Observation 2: The performance of Cecil releases is in terms of Cost per FP and Duration per FP, two to three times better than that of non-Cecil releases.

The quality of Cecil (0.13) and non-Cecil releases (0.14) in terms of *Defects per FP* is not as good as that of the best-in-class releases (Huijgens & van Solingen, 2013a), yet still within boundaries when compared to the overall score of *BelTel* as a whole (0.12). No indications are to be found in the quantitative figures that indicate too many defects during the development process.

Observation 3: Process quality in terms of Defects per FP of Cecil releases and non-Cecil releases is not as good as earlier described best-in-class releases from the full benchmark, yet on average when compared with BelTel overall.

Besides the three performance indicators we also calculated two metrics that give us an impression of the availability and reliability of the Divine system, based on the tickets that were made on failures in the production environment in the first two quarters of 2015. Based on 32 tickets the *Mean Time To Repair* (MTTR) was 6:35, and the *Mean Time Between Failure* (MTBF) was 107:12, indicating that on average once every 4.5 day a failure occurs that lasts for on average 6.5 hours. The most worrying signal is that during repair the Divine system usually is not available for end-users and only limited sales can be performed by all shops of *BelTel*.

Observation 4: The reliability and availability of the Divine system is worryingly bad and holds big risks for BelTel's business continuity.

Concerning the success factors we identified in earlier research (Huijgens & van Solingen, 2013a), we observe that three of them also apply to Cecil: a steady heartbeat, a fixed and experienced team, and release-based working on a single application. However, the factor of Scrum software delivery needs necessary differentiations.

The Cecil team certainly used a number of Scrum practices, such as a product owner, product backlog prioritization, and a product backlog management tool (*Scrumwise*) as a core instrument for its communication. And one may argue that although a product owner was distinguished, the availability and reliability of the Divine system was not included in this role, turning it into an information analyst with a Scrum label. Besides that, typical Scrum practices such as the role of the Scrum master and the daily standup meeting were not formalized within the team. Because of that we hesitate to label the Cecil releases as typically Scrum.

Observation 5: Three success factors identified in earlier research apply to Cecil too: a steady heartbeat, a fixed and experienced team, and release-based working on a single application. However, the Cecil releases cannot be defined as typical Scrum.

For all non-Cecil releases we observe that only the factor release-based working on a single application applies. There was no fixed team, experience was not secured once releases were finalized, and a plan-driven (waterfall) delivery approach was followed.

4.5. Results of the Interviews

In order to answer research question RQ4.2, 'what other factors can be found that influence release-based software delivery in a positive or negative way?' we performed nine interviews with eleven stakeholders. Table 4.4 gives an overview of the interview participants and their backgrounds. All interviewees are in one way or another involved in the Cecil releases, either as a user of

Table 4.4. Overview of Interview Participants.

	Role	Business / IT
P1	Business analyst CECIL releases and (part of) non-CECIL releases	IT-department
P2 P3	Team leader Billing & Rating Support Billing & Rating support agent	Business Business
P4	Product owner CECIL releases	Business
P5	Release manager	IT-department
P6 P7	Consumer Care Mobile team leader Consumer Care Mobile team leader	Business Business
P8	Tester CECIL releases	IT-department
P9	Team leader Roadmap & Release Management	IT-department
P10	Shop Support & Channel Communication team leader	Business
P11	Enterprise architect – former team leader CRM team (a.o. DIVINE system and CECIL releases)	IT-department

Participants are depicted in one row when a combined interview took place (e.g. P2 and P3 were interviewed together).

the Divine system (business), as a member of the Cecil team, as a stakeholder from IT release management, or as enterprise architect responsible for the application landscape.

Overall the interviewees are more or less satisfied with the Cecil releases, as illustrated by a statement of interviewee P2 who revealed that requirements in Cecil are cherished with a typical nickname ‘cecillekes’ [Belgian colloquial for 'sweet Cecilia'].

In the following Subsections we grouped aspects of observations from the interviews in nine categories.

4.5.1. Product owner is praised by many participants

The role of the product owner, and more specifically the way this role is fulfilled by the person in question, is in general highly appreciated; for stakeholders from business and IT alike. Many interview participants mention this as a first point when asked what aspects influenced the Cecil releases in a positive way. Examples are that the Cecil releases run well (P2), and that the Product Owner sets priorities within the backlog and determines the impact on the system (P1).

Observation 6: The role of the product owner and the specific personal fulfillment of that role is appreciated highly by stakeholders from both business and IT.

4.5.2. Cecil focuses on small but fast deliveries

Many stakeholders, especially those from business, mention as a success aspect that the Cecil approach focusses on quick wins; delivering high value for end-users (e.g. agents). Time-to-market is mentioned as a success factor (P5). We assume that the focus on delivering small enhancements in a fast and flexible way (P11, P1) is connected with this. Besides that, added value for the end-users (shops) is created by fast delivery with limited numbers of errors (P2).

'The idea of the Cecil items comes from the users. They see a bug in the system or a difficult process… I have to press this button twice or something… It is really based on ideas from the users. We discuss these ideas with [product owner] and see what can be put on the list.' (P3)

Observation 7: Cecil is about quick wins: small, fast deliveries of requirements based on end-user problems.

A remark here can be made on the fact that the Cecil releases are applicable to one single application, the Divine system. One interviewee (P1) mentioned this as a success factor; referring to the fact that Cecil releases are independent from other teams. This finding corresponds with an observation from earlier research on two teams in a similar setting in another company (Huijgens & van Solingen, 2013a).

4.5.3. Role of Scrum master is not formalized in practice

Contrary to that of the product owner, the role of the Scrum master is not formalized in the Cecil team. Despite the fact that at the start of the Cecil releases the team experimented with this role, it did not last in practice. Instead, the business analyst performs as an informal kind of coordinator in the team (P1).

'For me the most positive change was that [business analyst] joined the team. She replaced the former business analyst and she did a fantastic good job. She was just chasing IndSup-A, she was keen on getting feedback and followed-up what was open.' (P4)

Observation 8: The role of the Scrum master is not formalized in the team, yet no one seems to miss it.

4.5.4. Close cooperation within the Cecil team

Coordination within the Cecil team is an activity that is less formalized (e.g. there is no one with the formal role of Scrum master, coordinator, or project manager) yet the workflow seems to go smoothly with satisfied team members (P1). The transparent way of working together, and the open communication were mentioned as success factors (P7). The team members knew each other, lines were short, and the setting of the team was fixed and relatively small (P4). One time per week, or in the beginning even twice, a status meeting was organized (P1).

With regard to the fact that a part of the teamwork is done in India some remarks were made, although we did not get the impression that this was a large issue for the team itself. One Indian team member that works onsite acts as the main contact for the onsite team and as single point of contact for the offsite team members. Only small effects were observed here; handovers went quite smoothly (P4).

'An improvement was the replacement of [former developer from IndSup-A] by [actual developer of IndSup-A]. The former was most of the time in India and [actual developer of IndSup-A] is most of his time here onsite. So that was more difficult. More conference calls, the sound was very bad, I didn't understand the language too good. Now we do not have fixed meetings anymore. We just walk by when needed and if it's convenient we setup a meeting.' (P4)

Observation 9: Coordination is less formalized. It is a team activity; the Cecil team is a typical fixed and self-organizing team, although an onsite lead developer that coordinates offsite Indian team members helps a lot.

4.5.5. The Product Backlog management tool

A success aspect that is closely connected to the good cooperation within the team is the tool that is used for product backlog management: Scrumwise. Many interviewees mention it as very satisfying (P4, P5, P7). Stakeholders from business departments indicate that they use the tool for Kanban purposes in their own departments too. The tool supports good communication and bundles everything real-time together. People see right-away that someone is doing something (P4). It improves interaction between team-members and records all requirements and issues (5).

'Scrumwise is a good tool because it helps the team members to align activities. Everybody was available via the tool. How do we work? What are the agreements? It really had advantages for that purpose I think.' (P7)

Observation 10: The product backlog management tool Scrumwise positively affected communication.

With regard to documentation some small remarks were made that indicate that interviewees are satisfied with the level of technical design within Cecil, but that things need to be improved. However, this was experienced more as a general issue with regard to *IndSup-A* (P1). Remarks were mainly regarding improvements on version control and technical design activities (P7).

4.5.6. Improvement: Budget and Estimating is fuzzy

The budgeting process for the Cecil releases seems to be a bit fuzzy. Team members lack knowledge on what the budget is and how it is prepared. It is unclear how budgets are controlled and who is responsible for this (P4, P7).

With regard to estimation of new releases the same remark seems valid. The team does not use Story Points for estimating, but relies on estimates made by the developers of *IndSup-A*: during the planning process they strongly advise on what requirements are in or out of a release (P4).

4.5.7. Improvement: Testing

Three parties are involved in testing. Build and integration testing is performed by offshore testers of *IndSup-A*. UAT and regression testing is performed by a tester within the Cecil team from an external supplier that is performing company-wide test activities for *BelTel*. And finally testers from a Billing and Rating Support team perform a production test (P2).

Although all interviewees were unanimously satisfied with the Cecil process, some mention that testing can be improved, varying from smarter testing to follow-up of tickets recorded during the testing activities. Improvements were to be made in the somewhat informal, and laid-back approach on follow-up of test-tickets by *IndSup-A* (P4).

Yet, although some interviewees indicate that testing needs improvement, some are quite satisfied about the quality as delivered by the Cecil releases.

'If you compare Cecil with other projects then Cecil scores much better in numbers of defects per test case. The pre-project period as delivering requirements and test cases goes very smoothly. Also the requirements are described very well.' (P5)

Observation 11: Interviewees indicate that testing could be improved by smarter testing and better follow-up of test-tickets by developers of the Indian supplier.

4.5.8. Evolution of the process over time

All interviewed stakeholders are positive to very positive on the Cecil releases as they developed over time. Some interviewees even indicate that in case the Divine system is replaced in future, the process of enhancements as performed in the Cecil releases should be kept operational.

The team and its process is experienced as stable and people know where to go with questions (P2), and it delivers small enhancements, but with high impact for business stakeholders (P4). A remark is made on whether the scope of the team should be enlarged to also larger enhancements too (including requirements that were in scope of once only projects that acted on the Divine system too (P7).

'As far as we are concerned Cecil should go on like it is...' (P3)

Observation 12: All interviewed stakeholders are satisfied about the Cecil way of working and want to have it continued in future.

4.5.9. Bad performance issues of the Divine system

As described earlier in Section 4.2.1 the Divine system suffers from severe problems with regard to availability and reliability. Many interviewees relate to this by mentioning that the Divine system is a legacy system, at the end of its lifecycle, mainly due to high costs for system upgrades by its original supplier and due to the ongoing availability and reliability issues.

'The serious stability problems of Divine are especially owing to database administration and integration with the backend. Those are the two things that need an architectural adjustment.' (P11)

'The source of the performance problems is the fact that things are interwoven. If one application goes down most of the time fifteen others go down. Just replacing systems with the same functionality is only a good investment for the supplier and its partners. It's technical debt. But I call it also the blame game. We do a lot of system thinking instead of client thinking.' (P7)

'The reason that Divine is at the end of its lifetime is that we wanted to do an upgrade, but the costs from the supplier of the system were very high. The system is also not that young anymore. Together with the cost the decision was taken to go for another system. In a way Divine is built as a CRM system. Yet BelTel used it as a sales system with many customizations. It was not originally meant for that, and that's why there are many performance problems nowadays.' (P1)

'Divine is not very stable, we have large performance issues. Loading problems, or error messages. I think that's the most important reason to go to a new system. If you look at the high number of problems in the shops… They are talking to a customer and press a button and then they have to wait for two minutes… And all the Apache errors… The white screens when you have to completely log off and start again…' (P3)

Besides performance issues also the fact that the Divine system functionally evolved in a difficult to maintain solution for the business users is mentioned as a problem for the future.

'Functionally Divine works as it works… we made things wrong ourselves. We rebuilt things and Cecil could stick some plasters on some wounds. But there is no 'wow' to make out of it anymore…' (P4)

A decision is taken to replace the Divine system by a new product that will be a package off-the-shelf solution with minimal customization and minimal integration with backend systems.

'In former times Divine went down every day. Things improved. But do we have a proper CRM? No! That's why the system is going to be replaced in the coming six months. We are now looking at a new package solution with as little connections to Provisioning and Billing systems to make the dependencies as small as possible.' (P11)

Yet, it is interesting to observe that stakeholders tend to judge the Cecil releases and the Divine system as different things that are not interrelated.

'Cecil stands loose from the system that does not work. [Interviewer]: But why are there no performance improvement issues on the Cecil backlog? [Interviewee]: Well they tried all kind of things to improve the performance. I do not think that's going to be a quick win. A specialist came over from the USA to see how he could solve things. If Cecil was planned to solve these problems, then the management would already put performance things on the backlog.' (P2)

'This morning we had a meeting were the operations manager opened his heart on an issue last month, when Divine was down for one day due to firewall issues, and because of that BelTel did not make any money for a whole day. It is interesting why these non-functional aspects are not incorporated into Cecil. Apparently nobody thinks about this.' (P5)

'It is a classic problem within BelTel that Operations asks to be involved in a project... But effectively they only react per email on questions. We look at DevOps, but we are not even agile yet. What do you expect?' (P11)

Maybe the most striking finding of our analysis, is the fact that it is possible to have stakeholders that are all quite satisfied with the process of releasing a steady stream of enhancements over time, yet on the other hand they have to struggle with a software system that is lacking in reliability and availability.

Observation 13: A release process that performs better than average on cost and duration, and on average on defects, can satisfy stakeholders in managing changes on a badly performing software system.

4.6. Discussion

4.6.1. Threats to Validity

With regard to construct validity, the degree to which a test measures what it claims to be measuring a remark is in place on Function Point Analysis. Functional documentation is used to count FPs, holding the consequence that low quality documentation could have led to low quality FPA. To mitigate this risk, we thoroughly reviewed all sets on completeness and correctness, we made use of two certified FPA specialists, and assured that all involved FPA specialists are trained and experienced. To prevent from using low quality release data, we had all data reviewed by the product owner and the business analyst and the financial controller of *BelTel*.

By normalizing all project data with the functional size in FPs we warranted internal validity, the extent to which a causal conclusion is based on our study. By doing so we could more objectively compare performances of all releases in order to minimize systematic error. The effect of outliers is limited and the risk on bias is mitigated responsibly based on the diversity of projects and business domains within *BelTel*, the number of software projects, and the fact that we measured and analyzed software project portfolios as a whole in an empirical way.

To improve the internal validity of our study we performed a member check with participants. Although this was done in an informal way, we tried to improve the accuracy, credibility, validity, and transferability of the study by reflecting both during the interview process, and at the conclusion of the study, before results were presented and discussed with executive within the company.

With regard to external validity, whether the study results can be generalized to settings outside the study, we argue that due to the limited scope of our study, one specific series of releases in one company, it is too early to generalize the outcomes. Since we looked at seventeen releases performed in one company; the outcomes cannot be generalized to other environments without precautions. A promising factor here is however, that we compare the quantitative results of our study with results from existing research that we performed on similar software releases in another company, leading to a general expectation that the outcomes of our study might generalize to similar release approaches and companies too and that factors such as a steady heartbeat, release-based way of working mapped on a single application, and a fixed and experienced team are common success generators for software engineering.

4.6.2. Scrum as a Distinguishing Factor

Our research did not indicate that Scrum in itself is a distinguishing factor for the success of the software releases, but that some specific elements of Scrum, namely, the role of the product owner, a product backlog management tool, and short iterations based on prioritized requirements that deliver high value to end-users, are key elements that lead to release processes that outperform on cost and duration. The role of the Scrum master, daily standup-meetings, planning and estimations in Story Points, and the concept of Sprint Reviews were not adopted. As such, this setting should not be qualified as a Scrum setting, but as a local agile implementation using some Scrum practices. Furthermore, the fact that the Cecil team spends time on upfront tasks such as writing a design document deviates from many agile approaches.

4.6.3. Impact / Implications

With regard to our first research question we conclude that the Cecil releases corresponds to four of the five success factors mentioned in earlier research (Huijgens & van Solingen, 2013a), a steady heartbeat (taking into account that the duration of releases varied, but the release dates were preset upfront), release-based working on a single application, and a fixed and experienced team. The success factor Scrum did not fully correspond with the subject Cecil releases, but a number of Scrum practices were in place. The non-Cecil releases only correspond with one factor, namely release-based working on a single application.

Analysis of the observations from the interviews, based on a grouping of observations and connections in coherent categories, reveals three categories with regard to the Cecil releases: the Cecil team, the Cecil release performance, and the Divine system. Our study indicates that these categories apparently can live together without close connections. The Cecil team itself performs very well, stakeholders are quite satisfied, while the subject Divine system performs poorly in terms of both reliability- and availability.

For future use the model that we used to benchmark the performance of releases against our research repository, the *cost duration matrix*, should be expanded with metrics on the performance of software systems after deployment in a production environment. In this way concepts such as *good practice* and *bad practice* will reflect the performance of software releases in a more realistic way.

4.7. Related Work

Challenges with legacy systems as described in our case study are examined in many related work. (Boehm, 2006b) for example, mentions legacy evolution as one of the major future challenges for systems and software dependability processes. (Deursen et al., 1999) see “to try to bridge the gap between research aimed at building new software and research aimed at maintaining or renovating old software” as a large challenge.

A common idea of many research performed in the former millennium is that success and failure are interconnected with process-based activities (Hall et al., 2002). Reel, 1999) mentions five critical success factors in software projects, such as start on the right foot, maintain momentum, track progress, make smart decisions, and institutionalize post-mortem analyses.

(Niazi et al., 2006) identifies a large number of success factors for software process improvement implementation in existing literature. (Rainer & Hall, 2002) surveyed practitioners and found factors such as reviews, standards and procedures, training and mentoring, and experienced staff that practitioners generally considered had a major impact on successfully implementing SPI. Besides that, they found factors such as internal leadership, inspections, executive support and internal process ownership that the more mature companies considered had a major impact on successfully implementing SPI.

(Stelzer & Mellis, 1998) mention ten success factors of organizational change in software process improvement, a.o. management commitment and support, staff involvement, and providing enhanced understanding.

More recent work emphasizes the success and failure factors of shorter iterations due to an agile way of working. (Chow & Cao, 2008) surveyed agile professionals on success in agile software projects, and came up with factors such as delivery strategy, agile software engineering techniques, and team capability. (Misra et al., 2009) found factors such as customer satisfaction, customer collaboration, customer commitment, decision time, corporate culture, control, personal characteristics, societal culture, and training and learning.

(Sutherland et al., 2007) assessed hyper productivity in Scrum and mentions success factors such as team formation, Scrum meetings, sprints, product specification, testing, configuration management, pair programming, and measuring progress as typical for success.

(Meyer, 2014) identifies a number of contributions of the agile approach: refactoring, short daily meetings that support good team communication, identifying and removing impediments, and identification of sources of waste. As “brilliant agile principles” he mentions short iterations, continuous integration, the close window rule (no functionality can be added during an iteration), time boxing, the role of the product owner, an emphasis on delivering working software, the notion of velocity, and associating a test with every piece of functionality (Meyer, 2014). A remark on Meyer's observation on refactoring as a strong agile practice is that refactoring tools are seldom used by developers (Murphy-Hill, Parni & Black, 2012).

4.8. Conclusions and Future Work

In this study, we addressed the problem of different ways of working for evolving legacy software, which by definition resist change. The contributions of this chapter are threefold. First, it gives a description of a case in which a release-based, iterative process on a legacy system worked well, satisfying key stakeholders despite the poor quality of the system itself. Second, it confirmed three success factors as identified in our earlier research as contributors to this success (Huijgens & van Solingen, 2013a):

1. A steady heartbeat;
2. A fixed and experienced team
3. A release-based way or working, mapped on a single system.

Third, we identified four additional success factors:

4. The role of the product owner and the personal interpretation of that role;
5. A focus on quick wins and small, fast deliveries of requirements based on end-user problems;
6. The fact that the role of the Scrum master is not formalized, leading to a self-organizing team with an onsite lead developer that coordinates offsite Indian team members;
7. A specific product backlog management tool that positively influences communication.

The research as presented opens prospects for future research. With regard to our observation that the subject Cecil team applies less formalized aspects of Scrum in its process, we conclude it is important to examine whether the findings from this study are applicable to teams that work according to more formalized settings of Scrum too.

Finally, in interviews we heard stories about poor performance of the Divine system. Since we considered this to be out-of-scope for this study we checked this finding only quantitatively. For future studies we will include qualitative analysis of the underlying legacy system in our study too.

4.9. Acknowledgments

We thank *BelTel* for its generosity to allow us to use company data for our research, and the members of the Cecil team, and other interviewed stakeholders for their willingness and openness to share their experiences with us.

A statistical, evidence-based pricing approach for software engineering, as a single instrument, can be used in the subject companies to create cost transparency and performance management.

Photo by Zivile & Arunas

5. Evidence-Based Pricing of Project Proposals

Context: A medium-sized west-European telecom company experienced a worsening trend in performance, indicating that the organization did not learn from history, in combination with much time and energy spent on preparation and review of project proposals. *Objectives:* In order to create more transparency in the supplier proposal process a pilot was started on Functional Size Measurement pricing (FSM-pricing). *Method:* In this chapter we evaluate the implementation of FSM-pricing in the software engineering domain of the company, as an instrument useful in the context of software management and supplier proposal pricing. *Results:* We analyzed 77 finalized software engineering projects, covering 14 million Euro project cost and a project portfolio size of more than 5,000 function points. *Conclusion:* We found that a statistical, evidence-based pricing approach for software engineering, as a single instrument (without a connection with expert judgment), can be used in the subject companies to create cost transparency and performance management of software project portfolios.

This chapter was published as *Pricing via functional size: a case study of 77 outsourced projects* in the proceedings of the 9th International Symposium on Empirical Software Engineering and Measurement (ESEM) (Huijgens, Gousios & van Deursen, 2015c).

5.1. Introduction

This story is about a company that experiences two problems in its software engineering outsourcing. First, a worsening trend is seen in *project cost per function point*, indicating that the organization does not learn from historic projects. Second, much time and energy is spent on preparation and review of fixed price project proposals. Our case study explores whether a new project pricing method helps to solve these problems.

5.1.1. Problem Statement

To arrive at a price that is acceptable for both parties involved, most companies rely heavily on expert judgment (Jørgensen, 2004); where the advice of knowledgeable staff is solicited (Boehm, 1984). Usually this is performed as a bottom up approach, where component tasks are identified and sized and then these individual estimates are aggregated to produce an overall estimate (Boehm, 1984).

Yet, in practice effort and/or schedule overruns are business-as-usual (Moløkken & Jørgensen, 2003), despite involvement of experts. Software development is characterized by high cost and schedule overruns (Verhoef, 2002). Estimation errors are reported to be essential causes of poor management, due to lack of a solid baseline of size (Glass, 2002).

An alternative method for software project estimation is based on algorithmic cost models (COCOMO 2 is a well-known example) which take cost drivers representing certain characteristics of the target system and the implementation environment and use them to predict estimated effort (Boehm, 1984). In many of these statistical approaches size is assumed to be a key factor to estimate project cost (Abran et al., 2002a) (Gencel & Demirors, 2008). Usually size of software engineering projects is measured with a formal Functional Size Measurement (FSM) standard (IFPUG, 2009). FSM is a method to measure the size of software engineering projects by means of the functionality delivered to users (Gencel & Demirors, 2008), which lays the foundation for a statistical method of project pricing based on functional size. Advantages of such a statistical method are that this will help to improve transparency of estimations and that it can be a good instrument to create continuous improvement of project performance.

Although we did not find evidence in existing literature, our observation in industry is that a purely statistical method is almost never used. If statistical analysis is used, this is usually supplementary to an expert judgment-based approach (Jørgensen, 2004). And practice shows that in most cases the expert opinion – in many cases supported by reasoning by analogy – is leading when it comes to decision-making (Heemstra, 1992).

5.1.2. Research Objectives

The goal of this chapter is to examine whether a purely statistical approach to pricing is effective in an outsourcing context. We define an approach to be effective when a so-called win-win situation is achieved: meaning that both involved parties are satisfied and project proposals are perceived to be transparent for all stakeholders. The supplier delivers a service for a price that is higher than the cost, and the customer gets higher value than the paid price. In addition to that the outsourcing context asks for a long-term (5 year) relation. For this purpose we define three research questions:

RQ5.1: To what extent are both parties involved in the case study satisfied with FSM-pricing?

RQ5.2: To what extent does FSM-pricing help to improve transparency of project proposals?

RQ5.3: To what extent does FSM-pricing help to create cost and time improvements?

5.1.3. Context

In order answer these research questions, we performed a case study on the implementation and evaluation of FSM-pricing as a single instrument for software management, in a telecom company in Belgium (in this thesis indicated as *BelTel*), and the pricing approach agreed with its main Indian IT-supplier (indicated as *IndSup-A*). We studied data collected from 77 software projects that finalized during a period from 2012 to 2014. Moreover, we conducted 25 interviews including structured as well as open-ended questions. Our study is primarily descriptive, and not comparative in a sense that we compare outcomes with other companies: we do not have data to see how other pricing approaches might have worked. However, we do compare outcomes over time within the same company. Yet, we provide a rigorous analysis of what worked well, and what did not work well using FSM as an instrument for pricing.

The innovation of our study is that we raise the question to what extent a single, statistical, empirical approach to project estimation can reach the goal of transparent project proposals and due to that, cost and time improvements. The case study shows that FSM-pricing can successfully be used in the practice of *BelTel* and *IndSup-A*, as a statistical, evidence-based pricing approach for software engineering project proposals (RQ5.1), that FSM-pricing, in both subject companies leads to an improved transparency of project proposals and satisfied stakeholders (RQ5.2). Furthermore we found that FSM-pricing in our case study does lead on short term to cost improvements, but that no time improvements are realized within both subject companies: average project duration shortens, but average project size gets smaller too (RQ5.3). Due to the limited scope of the study it is too early to generalize the above mentioned findings to other companies and suppliers of software projects, yet we believe the outcome can help software companies to setup transparent and improving project pricing strategies.

We base the reporting structure of this case study on the linear-analytic structure as described in (Runeson et al., 2012). In Section 5.2, we survey earlier research on software pricing and discuss the background of FSM-Pricing. In Section 5.3, we chalk out the case study design. In Sections 5.5 and 5.6 we present results and we evaluate validity. In Section 5.7 we discuss the results and Section 5.8 includes conclusions and future work.

5.2. Related Work

When it comes to software pricing, two types of estimation techniques are distinguished to discover the cost of producing a software system; experience-based techniques such as expert judgment and algorithmic cost modeling where cost is estimated as a mathematical function of product, project and process attributes. A well-known example of the latter is Boehm's COCOMO 2 (Boehm et al., 2000a); more methods based on algorithmic software cost models with specific regression formula are widely used in industry, such as the Putnam Model (Putnam & Meyers, 2003), and SEER-SEM (Fischman, McRitchie, & Galorath, 2005).

Studies covered in a review by Moløkken and Jørgensen (2003) on *Surveys on Software Effort Estimation* mention a variety of estimation aids; such as work breakdown structure, Functional Size Measurement such as Function Point Analysis (FPA) (Gencel & Demirors, 2008), parametric tools (Lederer & Prasad, 1993), and qualitative methods (Bergeron & St-Arnaud, 1992).

For a long time researchers and practitioners have been investigating the use of statistics in software estimation. Since the 90s a limited number of studies has been published on the subject of pricing of projects based on statistics (Dekkers & Forselius, 2010) (Czarnacka-Chrobot, 2010). Despite all models and practices actual software estimation seems difficult. (Moløkken & Jørgensen, 2003) observe that 60-80% of the projects encounter effort and/or schedule overruns. Estimation methods in most frequent use are expert based: expert consultation, intuition and experience, and analogy. Frequent use of expert judgment is advocated because of a lack of evidence that formal estimation models lead to more accurate estimates (Moløkken & Jørgensen, 2003).

Although research in the field of software engineering often shows conclusion instability (where what is true for project one, does not hold for project two) (Menzies & Shepperd, 2012), and expert judgement is common practice, studies do emphasize pitfalls. (Jørgensen & Gruske, 2009) argue that estimation professionals in many cases do not use lessons learned from finalized projects. (Valerdi, 2011) mentions cognitive bias that can make experts produce poor estimates. (Passos et al., 2011) show that many experts generalize from their first estimates to future ones. Recent literature study on agile metrics shows high popularity of velocity for effort estimates in industrial agile teams (Kupiainen et al., 2015); yet, cost metrics and size related metrics, and especially metrics related to pricing of projects, are not mentioned. (Fink & Lichtenstein, 2014) address the gap between project size (although measured here in cost and not in functional size) in the software engineering literature and the attention it receives in software contracting research. (Madachy et al., 2011) argue that due to impreciseness of general software cost parameters such as size, effort distribution, and productivity cost database better are segmented by domain.

(Abran et al., 2014) uses a FSM-based model to assess productivity and to estimate new projects on fixed and partly variable costs. (Ramasubbu et al., 2011) reveal complex tradeoffs in choosing configurational choices that are optimized for productivity, quality, and profits. A discussion on model-based versus judgment-based is described in (Benestad & Hannay, 2011), indicating a substantial overlap between the two approaches, but also some mismatches.

We did not find studies that describe dedicated use of algorithmic cost models in practice, without interference of expert-judgment based methods.

Limited research is performed specifically on the topic of pricing software projects. We have not found any studies that emphasize the use of FSM as a single instrument for pricing. This is remarkable; several studies on FSM stress that software size is a primary predictor of project effort and thus project cost (Gencel & Demirors, 2008) (Abran, Silva, & Primera, 2002b).

5.3. Case Study Design

5.3.1. Theory

FSM and FPA

FSM is an industry standard to measure size of software engineering activities. Five FSM methods are certified by ISO as an international standard; in our study IFPUG FPA (ISO 2003c) (IFPUG, 2009) is used. FSM origins from FPA, designed by Albrecht (1979) to estimate size of software delivery by means of user functionality. FSM is based on the complete set of functional requirements of a software project. An extensive overview of FSM can be found in (Gencel & Demirors, 2008).

FSM-pricing

FSM-pricing, as used in the context of this case study, is a method that we developed for pricing of proposals for software projects to be performed within *BelTel*, by *IndSup-A*. In order to define a fixed price for a project, first FSM is performed to measure the functional size of a project, second the price of the project is determined based on a power trend that is built on historic data of finalized software projects. In our case study we only used historic data of projects that were finalized within the practice of *BelTel* and *IndSup-A* itself. The FSM-pricing method is explained more in detail in Subsection 5.3.5.

5.3.2. Research Questions

In the period prior to FSM-pricing become operational within *BelTel*, we discovered two major disadvantages in the current expert-judgment-based estimation approach through analysis of finalized software engineering projects. First, *BelTel* showed a worsening trend in project cost per FP, indicating that the organization did not learn from historic project data. Second, much time and energy was spent on preparation and review of fixed price project proposals, leading to long project durations. To turn the tide on the worsening cost and time performance, and to smoothen the proposal process, a decision was made to change towards an empirical, evidence-based, and analytical way of preparing fixed price project proposals. FSM-pricing was born, having two goals, defined by *BelTel*'s management: 1) improve transparency of proposals, and 2) create ongoing cost and time improvements of software delivery due to the expected improved clarity in the delivery process (e.g. less discussion on cost and scope).

Based on this we defined three research questions, with the intention to find out to what extent stakeholders involved in FSM-pricing are satisfied about the method, to what extent the method helps to improve transparency of project proposals, and to what extent cost and time improvements are realized.

5.3.3. Case and Subject Selection

FSM-pricing, as described in this chapter, was implemented in the software project department of *BelTel*, as part of a transformation program that includes a change from one large European IT-supplier to a large Indian IT-company (*IndSup-A*) for the majority of its software engineering activities for the Customer Relationship Management (CRM), Billing, and Data Warehouse (DWH) applications. Besides the fact that a 5-year sourcing contract was agreed between *BelTel* and *IndSup-A*, both companies were not in any way - besides contractually - related. FSM-pricing aims to implement FSM based on FPA (IFPUG, 2009) as an approach to improve the capability of the company to challenge *IndSup-A*'s proposals for to-be-started software engineering activities. All proposals were fixed-price; no extra time-material cost were allowed unless the scope of a project (in FPs) was changed during the delivery period.

Based on this organizational definition, and driven by the goal to investigate a representative subset of mutually highly different software projects within a company's software portfolio as a whole, we decided to select all software projects to be finalized during the period January 2014 to December 2014, within the business domains CRM, Billing, and DWH of *BelTel*, with *IndSup-A* acting as the main supplier, to be subject of our case study. For benchmarking purposes we used a subset of historic software projects that were finalized in the period 2012 to 2013, within the three business domains of *BelTel*, yet performed by other external suppliers than *IndSup-A*.

5.3.4. Data Collection procedures

Data of all software projects that are collected are measured by a team of *BelTel*, supported by measurement specialists of *IndSup-A*. The author of this thesis was leading *BelTel*'s measurement team during the case study. As a source for the project data we use the formal project administration. All project data is reviewed by the applicable project manager and the financial controller of *BelTel*, and adjusted where needed. We collect both quantitative data (e.g. core metrics such as size, effort, cost, duration) and qualitative data (e.g. project backgrounds, factors that influenced a project) in a measurement repository. Projects cover a mix of the business domains CRM, Billing, and DWH, project types (e.g. newly built systems, enhancements, off-the-shelf packages), and project sizes (e.g. small enhancements, large once-only projects). In all projects the design, build, and testing activities are performed by one or more external suppliers. Most software projects are combined in releases and delivered at one moment to the business organization; each year eight releases are rolled out under guidance of a portfolio management team of *BelTel*.

We collect data on finalized software engineering projects only; stopped or failed projects are not included in our case study. We exclude projects that are only about infrastructure, or that include only non-functional requirements (e.g. performance, security), because these were not to be counted in FPs.

For all to-be-analyzed software engineering projects, we measure project size in Function Points (FPs), according to FSM ISO/IEC 20926 guidelines (IFPUG, 2009). FPA is performed by specialists either from a *BelTel* measurement team (in the period that *IndSup-A* is not in scope as main supplier yet), or by a *IndSup-A* measurement team (once *IndSup-A* is in scope as main supplier they perform all FPAs). Every FPA is reviewed on correct utilization of counting practices by an experienced IT-metrics expert who is also the author of this thesis, and on correct interpretation of requirements by an applicable subject matter expert of *BelTel*.

5.3.5. Analysis Procedure

In order to test whether cost or time improvements are realized we calculate the following performance indicators for each project (we opted for this set of indicators because they were included in the standard set of KPIs within *BelTel* and therefor to be expected as known by both parties management):

1. *Project cost per FP*: total project cost divided by the project size, expressed in Euros per FP;
2. *Build & Test cost per FP*: cost of the Build & Test phase divided by the project size, in Euros per FP;
3. *Project duration per FP*: duration of the project from start of the Initiation phase to technical go live divided by the project size, in Days per FP.
4. *Build & Test duration per FP*: duration of the Build & Test phase divided by the project size, in Days per FP.

When in this study *cost per FP* or *duration per FP* is mentioned without any prefix, the project version of each indicator is meant, instead of the Build & Test version. For analysis purposes results of individual projects are aggregated to company level, where project size (FPs) is used as weighting factor. All data used in the analysis were shared and thoroughly reviewed by measurement experts of both *BelTel* and *IndSup-A*.

Based on analysis of projects performed by *IndSup-A*, we calculated two domain-specific baselines on build & test cost per FP; these were going to be the trend lines for FSM-pricing. To create the baseline, we obtained the best fit after conducting a log-log transform. After performing a power regression, the resulting price calculation formula is:

$$Price = a * (\mathrm{FP})^{\beta}$$

The coefficients α and β may differ per application domain. In the portfolio under study, we typically have $\beta \approx 0.75$ (this value was based on analysis of the two baselines created within *BelTel*). Note that this formula is in line with COCOMO 2's effort estimation formula (which uses KLOC instead of function points) (Boehm et al., 2000a). We use simple regression on project size and build & test cost with power fit. Our foundation of this argument is that such a model facilitates greater analyzability and thus helps improving transparency.

For a statistics-based explanation we create a cross correlation table to determine, and filter the strongly dependent variables in our sample out from the regression model. We found that size and duration are all pair-wise highly correlated; we rejected duration and only used size as a predictor for cost. See the technical report for more details on statistics (Huijgens et al., 2014b).

We prepared two baselines: 1) CRM/Billing ($R^2 = 0.5621$) and 2) DWH ($R^2 = 0.9048$). CRM/Billing domain projects are combined in one baseline because the analysis shows no large differences between projects from both domains, many projects overlap domain borders, and because not enough data were available for proper individual trend lines for both domains. A separate DWH baseline was setup because these projects show a different pattern. See the technical report (Huijgens et al., 2014b) for plotter charts and details on the setup of both baselines.

Based on both baselines a tool was set up for cost calculation in project proposals by *IndSup-A*. For all to be started software projects the fixed price is calculated with this tool. Once the size of a project is counted and reviewed, the tool calculates the price for a project to be performed by *IndSup-A* based on the applicable domain baseline.

Stakeholders from *BelTel* opted strongly for a single pricing approach (only based on statistics), because ongoing discussions on project estimates were expected due to a variety of expert opinions if two approaches were to be used simultaneously, and because of that longer project durations. To reassure stakeholders of *IndSup-A* with doubts on this single method for supplier proposal pricing, a six month's FSM-pricing pilot was started. This pilot is the subject of the case study that is discussed in this chapter. Quantitative analysis is performed over the scope of the six-month pilot and the following six months operational use of FSM-pricing.

5.3.6. Model Validation Procedure

In order to validate the FSM-pricing method we use a mixed methods methodology, as we are examining a phenomenon with multiple (qualitative and quantitative) tools. We perform a single-case, holistic case study that involves two instruments; a survey consisting of open and closed questions, and a quantitative analysis of actual project data. The survey is performed six months after the start of the case study, the quantitative analysis is performed at the end of the case study period of one year.

To answer RQ5.1 (*To what extent are both parties involved in the case study satisfied with FSM-pricing?*) and RQ5.2 (*To what extent does FSM-pricing help to improve transparency of project proposals?*) we create a combined 10-minute questionnaire survey. The survey topics and the survey approach are determined in a number of preparation sessions between management representatives and the measurement experts of both *BelTel* and *IndSup-A*. Our aim is to come up with a manageable set of topics that would represent the pilot effectively. The survey consists of a number of closed questions; respondents are asked to rate these survey topics on a 5-point Likert scale. Next to the 5-point scale for each of the survey topics a choice of "Don't Know" as an answer is an option. Besides that the survey contains three open questions.

The survey starts with the collection of demographic information, and the answering of two partially closed questions: "What company are you working for?" and "What is your connection with FSM-pricing?" Both questions are intended to find out any differences in satisfaction with FSM-pricing within both the involved parties *BelTel* and *IndSup-A*, and between respondents with different roles. A comprehensive overview of setup and respondent statements in the survey can be found in the technical report (Huijgens et al., 2014b).

To assess the experienced satisfaction with FSM-pricing we asked respondents to answer the question *"How satisfied are you with the following?"* respondents are asked to rate 14 survey topics. To find out whether respondents feel that FSM-pricing needs to be continued a question is asked to be answered with yes or no: *"Should FSM-pricing be continued as an operational practice once the pilot is finalized?"* To understand possible reasons behind the closed questions we ask the stakeholders to answer three open questions (max 3 answers are allowed):

1. What is going well during the FSM-pricing pilot that we want to continue?
2. What is not going well during the FSM-pricing pilot that we want to fix?

In order to assess the experienced transparency with regard to project proposals we perform a survey with eight closed questions. The first seven (Q01 to Q07) are intended to find out how respondents experience the quality of artifacts and processes with regard to FSM-pricing. As a response to the question *"How would you rate the quality of the following?"* respondents are asked to rate these seven survey topics. Next to these questions three additional questions (E01 to E03) are asked: *"To what extent did you experience a change on…?"* respectively the transparency of proposals during the FSM-pricing pilot, the project cost per FP measured in euros per FP and the project duration per FP measured in days per FP.

RQ5.3 (*To what extent does FSM-pricing help to create cost and time improvements?*) is answered by performing quantitative analysis of project data. We analyze the performance of 77 finalized software engineering projects. For our study we use data of three categories of software engineering projects, all performed within *BelTel*:

1. *Repository*: data of historic projects in the period preceding FSM-pricing, not performed by *IndSup-A* (n = 22);
2. *Baseline*: data of finalized projects performed by *IndSup-A* used to prepare the FSM-pricing baseline (n = 16);
3. *Pilot*: data of finalized projects performed during the pilot that are in scope of FSM-pricing (n = 10);
4. *Operational*: data of projects finalized during the six months following the pilot (in scope of FSM-pricing) (n = 29).

In order to benchmark the outcomes of the qualitative analysis with industry peer groups we use a research repository of 331 comparable projects from other companies that we collected in earlier research (Huijgens et al., 2014c). All compared peer group projects from this benchmark repository conducted software engineering in business environments. Peer group projects were measured, collected, and recorded in the same way as conducted in this case study.

5.4. Results

5.4.1. Case and Subject descriptions

In this section we report results based on the three research questions of our study. We sent 41 survey requests by email to 17 employees of *BelTel* and 24 employees of *IndSup-A*. We selected these stakeholders because they are all involved in the FSM-pricing pilot. Twenty seven (27) surveys are returned, of which 2 are assessed to be incomplete (respondents only noted that they knew too little of the subject). 25 surveys are completed (completion rate 61%); the

Table 5.1: Backgrounds From Survey Respondents

Respondent background	BelTel n=11 (44%)	*IndSup-A* n=14 (56%)
Overall IT-management	28%	29%
FPA Measurement Team	18%	14%
Portfolio Management	27%	0%
Data Warehouse Team	9%	14%
CRM/Billing Team	9%	36%
Other	9%	7%

analysis in this study is based on these completed surveys only. Table 5.1 summarizes the backgrounds of the respondents that completed the survey.

Besides the results of the survey ratings we collected a large amount of open ended text from our survey. The first open question *"What is going well during the FSM-pricing pilot that we want to continue?"* resulted in 46 answers. The second open question *"What is not going well during the FSM-pricing pilot that we want to fix?"* resulted in 47 answers. 44 Answers were given to the question "What can we do to improve FSM-pricing?" In total 2,007 words were produced.

In this section we label respondents as P1 through P25 and we include results from the open text analysis where applicable. To analyze the free text answers, we adopt the coding technique described by (Runeson et al., 2012).

We apply high level codes and medium level codes and count the frequency of each code. A summary of the results of this analysis is shown in the following Subsection.

5.5. Results of the Qualitative Analysis

As is common in case studies, answers on surveys contain a substantial element of narrative. As these are representatives of the complexities and contradictions of real life, we include a selection of statements made by the survey respondents in the section on open ended text analysis in our study. We try to include examples of respondent statements that apply to differences as well as similarities.

Table 5.2: Survey Results (left part of the table).

Nr	Survey Topic (How satisfied are you with the following?)	Mean Overall
S09	Function Point Analysis method (IFPUG, estimated count)	3.96
S02	FSM-pricing pilot period itself	3.87
S01	Preparation of the FSM-pricing pilot	3.75
S15	Overall FSM-pricing	3.72
S13	Advantages of FSM-pricing for BelTel	3.68
S07	Pricing table for DWH	3.50
S12	Proposal Process (with regard to FSM-pricing)	3.42
S04	Management Commitment on FSM-pricing	3.42
S14	Advantages of FSM-pricing for IndSup A	3.40
S03	Communication with regard to FSM-pricing	3.39
S06	Setup of the IndSup A Baseline	3.30
S08	Pricing table for CRM / Billing	3.28
S05	Reliability of the FSM-pricing	3.28
S11	Coverage of FSM-pricing	3.26
S10	Waiver procedure for Function Point Analysis (exclusions)	3.25
Nr	Survey Topic (To what extent did you experience change on...?)	
E01	Transparency of Proposals	3.88
E02	Project Cost per FP (Euros per FP)	3.33
E03	Project Duration per FP (Days per FP)	3.00
Nr	Survey Topic (How would you rate the quality of the following?)	
Q02	Function Point Analysis performed by IndSup A	3.83
Q03	Function Point Analysis Review by Company C	3.78
Q07	The Overall FSM-pricing method	3.64
Q06	The IndSup A Proposals based on FSM-pricing	3.52
Q05	The CRM / Billing Baseline used for FSM-pricing	3.47
Q01	Requirements delivered by Company C	3.44
Q04	The DWH Baseline used for FSM-pricing	3.43

Sorted by Mean Overall; higher is better.

Survey Results (right part of the table)

Standard Deviation	Mean Company C	Mean IndSup A	Effect Size Company C / IndSup A	Effect Size Management / Development
0.81	4.00	3.92	0.08	0.11
0.55	3.91	3.83	0.08	-0.20
0.90	3.82	3.69	0.13	0.00
0.74	3.64	3.64	0.00	0.08
0.65	3.80	3.58	0.22	-0.30
0.73	3.86	3.22	0.63	0.15
0.88	3.70	3.21	0.49	0.06
0.83	3.64	3.23	0.41	0.25
0.68	3.29	3.46	-0.18	0.18
0.66	3.36	3.42	-0.05	0.22
0.93	3.55	3.08	0.46	0.13
0.83	3.57	3.09	0.48	0.22
0.94	3.55	3.07	0.47	0.09
0.92	2.70	3.69	-0.99	-0.45
1.03	3.00	3.46	-0.46	0.38
0.65	3.82	3.93	-0.11	0.36
0.70	3.40	3.29	0.11	0.17
0.76	2.78	3.15	-0.37	0.42
0.70	3.70	3.93	-0.23	-0.06
0.60	3.73	3.83	-0.11	-0.11
0.57	3.55	3.71	-0.17	-0.22
0.65	3.55	3.50	0.05	0.12
0.80	3.57	3.40	0.17	-0.05
0.65	3.45	3.43	0.03	-0.01
0.76	3.71	3.14	0.57	0.55

Table 5.2 summarizes the survey results. The two last columns show Effect Size calculated as two measures: 1) for each survey topic the difference between the mean *BelTel* score and the mean *IndSup-A* score, and 2) for each survey topic the difference between the mean Management score (all scores of respondents with the profile Overall IT-management, FPA Measurement Team, Portfolio Management, and Other) and Development (all scores of respondents with the profile Data Warehouse Team, and CRM/Billing Team). A negative Effect Size indicates *BelTel* / Management respondents are less satisfied with a survey topic than *IndSup-A* / Development respondents. A positive Effect Size indicates *BelTel* / Management respondents are more satisfied with a survey topic than *IndSup-A* / Development respondents.

We found the following with regard to satisfaction with FSM-pricing based on analysis of the survey results (Table 5.3 inventories an overview of the results from the qualitative analysis).

5.5.1. 88% want FSM-pricing as operational practice

On the question "Should FSM-pricing be continued as an operational practice once the pilot is finalized?" 80% answered "Yes"; 8% answered "Ok, but with improvement points (e.g. include effort of non-functional requirements").

5.5.2. FPA is appreciated by both parties

Both *BelTel* and *IndSup-A* respondents appreciate the applied FPA method (IFPUG, estimated counts); based upon the highest overall mean score of the survey (3.96). Besides that both parties appreciate the quality of the function point analyses that are performed by *IndSup-A* (3.78), and the reviews done by *BelTel* (3.80).

Qualitative analysis confirmed this finding. Many respondents considered the quality of the FPA high:

'Good Function Point review by BelTel and IndSup-A FPA-teams before proposal submission. (P10). Appreciate the way Function Point counting is done by IndSup-A.' (P23)

Many remarks made by respondents were related to requirements; which makes sense since requirements usually are the basis for project proposals. A noteworthy side-effect of FSM-pricing is that respondents experienced an improvement of the requirement management process during the pilot.

Table 5.3: Summary of the Open Ended Text Analysis

Category Name / Medium Level Code
Interactions, communications, people
Improved proposal transparency
Improve knowledge of Function Point Analysis and FSM-pricing
Discussion on size when lower price is expected or on waivers
Organization, processes
Uniform, standard and simplified process
Too small projects; no focus on release-based working
Delay due to search for clarity and review
Improve pricing tables (e.g. benchmarking, more realistic figs.)
Promote release-based working based on size
Promote pricing tables based on applications (technology)
Measurements
Perform gap-analysis on FSM-price versus actual effort spent
Requirements
FSM-pricing does not cover non-functional requirements
Low reliability of FSM-pricing when compared to actual effort
Improved Requirement Management
Artifacts
Good quality of Function Point Analysis process and products

Most of the details are sorted out at the time of proposals. Earlier these details were discussed in design phase. (P17)

'The solution is looked into more detail in order to get the right Function Points at the proposal stage itself. This helps in early detection of issues and resolution.' (P2)

This positive effect on requirements management might even be one of the main reasons for FSM-pricing success.

5.5.3. *BelTel* management: coverage needs improvement

Coverage is about the number of projects in *BelTel*'s IT-portfolio that is subject of FSM-pricing. Based on a relatively low mean value for *BelTel* (2.70), combined with an Effect Size of -0.99 between *BelTel* and *IndSup-A*, we conclude that respondents from *BelTel* are more than average dissatisfied about the coverage of FSM-pricing. An Effect Size of -0.45 between Management and Development indicates that coverage is a management rather than a developer concern.

We conjecture a connection with low rating of the waiver procedure by *BelTel* respondents; this procedure allows *IndSup-A* to exclude a project from FSM-pricing. A standard waiver is applied for infrastructure projects, configuration projects, and projects executed by other external suppliers. Also qualitative analysis revealed indications that ongoing discussions tend to be related with waiver requests:

Many ongoing discussions on waiver requests occur. (P20)

5.5.4. IndSup-A development: reliability needs improvement

In the context of FSM-pricing by reliability we mean whether respondents experience the outcome of FSM-pricing to be in line with their own judgment. *IndSup-A* developers seem dissatisfied with FSM-pricing where it comes to reliability. Proposal process (Effect Size 0.49), both pricing tables (0.48 and 0.63), reliability of FSM-pricing (0.47), and setup of baselines (0.46) are all rated low. We believe these are connected, but we did not find evidence for this in our data.

Looking at this aspect further in the qualitative analysis shows a feeling of disagreement between the outcome of FSM-pricing and effort-based estimates. Many respondents, especially from *IndSup-A*, mention that FSM-pricing does not cover Non-Functional Requirements and complexity (technology).

'FPA is not applicable to projects where more testing efforts are required for less development changes.' (P5)

'All the projects do have different non-functional requirements or technology; due to this the efforts differs.' (P2)

'The complexity of the changed code does not match with the amount of functionality to be changed, causing a disparity.' (P16)

We identified one specific measurement-related issue: the wish to perform a gap-analysis to find any differences between FSM-pricing proposals and actual effort spent in a project:

'To keep the counting simple we are considering all the requirements are at average level; we may need to perform gap analysis if the requirements mix is really averaging out on efforts.' (P17)

'Cross verification with actuals towards the end of project to revalidate the estimates would be an improvement.' (P7)

We identified a need for gap-analysis in order to identify differences between (estimated) project cost and actual effort. We consider conducting this gap-analysis as future research. With regard to the experienced transparency of project proposals we observed one major finding:

5.5.5. 84% experienced improved proposal transparency

Many respondents experienced an improvement of the transparency of project proposals during the FSM-pricing pilot (72% said transparency improved; 12% said greatly improved). Qualitative analysis confirmed this finding. Respondents mention improved transparency as a positive outcome of the FSM-pricing pilot:

'A good point is that there is less discussion.' (P8)

Some respondents see improved transparency as a driver for better requirements or to solve disagreements between customer and supplier:

'Instead of plain list of entities that we were maintaining in work-break-down-structure entities, we now have clarity on what kind of functionality is getting delivered.' (P17)

'Function points analysis sometimes is a constructive argument in case of disagreement.' (P20)

We observed the fact that FSM-pricing is experienced as a uniform, simplified process is on top of respondents' list:

'FSM-pricing is a single point for the final estimation, answerable to all stakeholders. The estimation review process becomes very simple. A standardized process, which can be trusted from both vendor and client stakeholders.' (P24)

'Uniformity in pricing approach as it does not depend on individual components to derive their efforts.' (P2)

'Avoid delays and budget overruns as estimation can be done at an initial stage against task-based.' (P13)

Table 5.4: Performance over three Project Categories.

Performance Indicator	Repository	Baseline	Pilot
Number of projects (n)	22	16	10
Average project Size (FP)	157	183	25
Project Cost per FP (EUR/FP)	2,651	1,485	2,560
Project Duration per FP (Days/FP)	2.35	1.58	7.17
Average project Duration (Months)	12,11	7,53	7,38
Performance Indicator	Rp	Bl	Pi

5.6. Results of the Quantitative Analysis

Data from three categories of 77 software engineering projects are used for quantitative analysis of project data (resp. *repository*, *baseline*, and *pilot*). In Table 5.4 we summarize the performance indicators for these three project categories. The analysis resulted in the following findings:

5.6.1. Project Duration per FP not in sync with peer groups

Analysis of the performance of the software engineering projects of *BelTel* shows that, although the project cost are in line with the prevailing market, the organization suffers from project durations that are substantially longer than those of peer groups in industry. An external benchmark against historic data of 331 finalized software engineering projects (Huijgens et al. 2014c) from different companies shows that a majority of the finalized projects of *BelTel* are cost effective (average *project cost per FP* is 46% better than the peer groups, see Figure 5.1), yet project durations are longer than the average of the total research group: average *project duration per FP* is more than twice that of the peer groups, see Table 5.5. This finding is applicable to all four categories of software projects performed within *BelTel* in our research repository, yet *project duration per FP* is worsening during the pilot.

We plot both all *BelTel* and peer group projects in a *cost duration matrix* (see Figure 5.1) (Huijgens et al. 2014c) (see Subsection 2.3.2 for a description of the *cost duration matrix*). This matrix shows for each project the measure of deviation from the average trend line (average of peer group projects plus

Table 5.5: Performance compared to Peer Groups

Performance Indicator	Company C	Peer Group	Delta
Number of Projects (n)	26	331	n.a.
Average Project Size (FP)	126	261	-52%
Project Cost per FP (EUR/FP)	1,604	2,983	-46%
Average Project Cost (K Euro)	203K	780K	-74%
Project Duration per FP (Days/FP)	2.20	1.04	112%
Average Project Duration (Months)	9,14	8.92	2%

Performance of Company in comparison with peer group projects from our research repository. Only finalized projects that were performed by INDSUP A are incorporated.

BelTel projects) expressed in a percentage; negative when below the average trend line, positive when above the trend line.

The matrix is divided in four quadrants. Each quadrant is characterized by the measure of negative or positive deviation from the average trend. When analyzed it shows that 80% of the projects is assessed to have a longer than average duration. 25% of the projects are in the *bad practice quadrant*; these projects perform in both cost and duration worse than average. 55% ends up in the quadrant *cost over time*; costs are less than average, yet project duration takes longer than average. Due to these deviating percentages we argue that Company A's *project duration per FP*, measured in *days per FP*, is not in sync with its peer groups; *BelTel* should improve its *project duration per FP* in order to stay competitive in the market.

Our analysis is that the bad *project duration per FP* is caused by two problems. First, the combined release approach of *BelTel* causes waiting time (waste) and unnecessary dependencies between projects. Second, average project duration conform industry, yet combined with small average project size cause a bad *duration per FP* as illustrated in the following.

5.6.2. Small projects block improvement

A finding with regard to project size is that from 2013-Q3 onwards substantially more very small projects (e.g. with a project size less than 30 FPs) are performed. We did not find any reason that could explain this reduction of

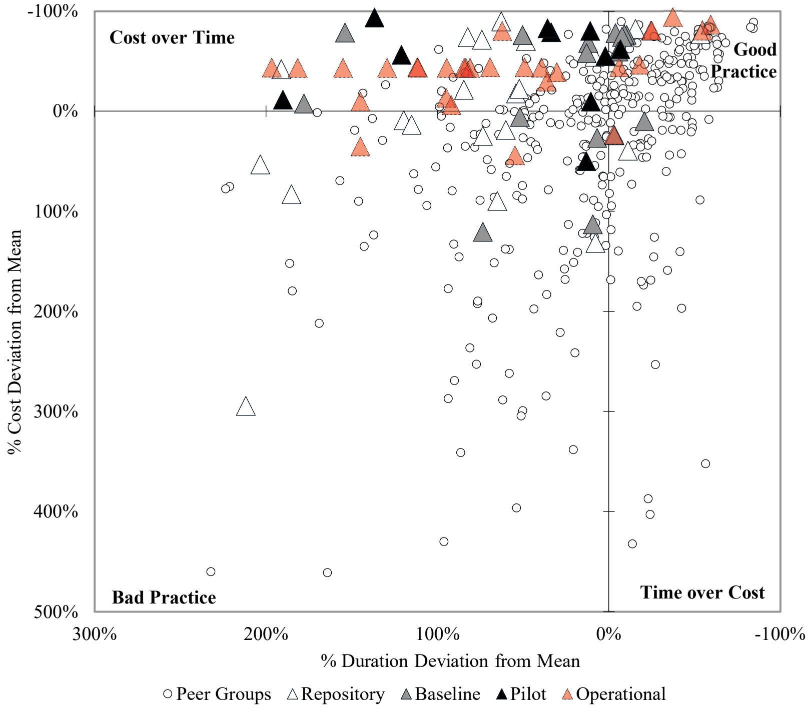


Figure 5.1: A cost duration matrix showing results of the quantitative analysis.

project size. Although smaller projects are from a cost point of view advantageous for *IndSup-A*, portfolio managers of *BelTel* are responsible for the construction of a specific release portfolio (a number of projects combined in one release; to be delivered at one specific moment). The idea that small projects from an economy-of-scale perspective should be combined is mentioned by some respondents in the open ended text as well:

'IndSup-A divides the offer in small pieces; we must have release based funding to make use of economy-of-scale.' (P8)

'Too many small projects are negative for BelTel due to economy-of-scale effects.' (P3)

We observed that in 2014 the throughput (total delivered number FPs) is approximately 29% lower than in the preceding years (see Table 5.6). One can argue that the maybe rather rigid approach of FSM-pricing is not sufficiently

Table 5.6: Performance over time.

Performance Indicator	2012-2013	2014	Delta
Number of projects (n)	38	39	n.a.
Average project Size (FP)	168	68	-59%
Throughput (FP)	6,366	2,660	-29%[1]
Project Cost per FP (EUR/FP)	2,116	1,679	-21%
Project Duration per FP (Days/FP)	2.00	3.52	76%
Average project Duration (Months)	11,69	7,90	-25%

[1] Throughput percentage is calculated based on extrapolation per year.

encouraging for *IndSup-A* due to a somewhat single-sided focus on cost reduction. However, *BelTel* promotes the idea that delivery of more throughput where applicable is desired. Looked upon from this side FSM-pricing underlines the delivery of more value for less money; and at the same time it rewards throughput enlarging by creating more turnover for the supplier.

5.6.3. Cost improves; yet, Duration does not

Looking at cost and duration over time (see Table 5.6) we find that *cost per FP* (the *cost per FP* measured over the whole project lifecycle from initiation to technical Go Live) improves by 21% in 2014 onwards compared to the years before. However, *duration per FP* is not. Next to our finding that *duration per FP* is substantially higher than that of the peer groups, no sustained improvements with regard to project durations are seen when assessed over time. *Duration per FP* shows a worsening trend. As discussed before the small size of many projects and the amount of waste in projects plays an important role here.

5.7. Discussion

Analysis with regard to RQ5.1 (*To what extent are both parties involved in the case study satisfied with FSM-pricing?*) resulted in four findings. First, 88% of the respondents of our survey want FSM-pricing as an operational practice once the FSM-pilot is finalized.

Second, the applied method for FPA, including the counting itself as performed by and *IndSup-A* and the review by *BelTel*, is appreciated highly by both respondents of both parties.

Third, coverage of FSM-pricing with regard to *BelTel*'s IT-portfolio is experienced as to be improved, mainly by managers from *BelTel*. Additional analysis of the measure of coverage of FSM-pricing with regard to the IT-portfolio shows that at finalization of the FSM-pricing pilot 27% of all IT-portfolio costs were calculated based on FSM-pricing. At the end of the operational period (end 2014) the coverage was improved to 52%. The remaining 45% is among others related to infrastructure (19%), support (17%), third party projects (5%) and small innovations (3%).

Fourth, developers from *IndSup-A* are dissatisfied with the reliability of FSM-pricing. The major reason for this seems to be that they experience little possibilities to incorporate non-functional requirements and complexity in project proposals. From a statistical point of view all projects are treated as average, where non-functional requirements and related complexity are incorporated in both trend lines.

To finalize our discussion on RQ5.1; an additional positive signal with regard to this is that after evaluation of the FSM-pricing pilot both *BelTel* and *IndSup-A* agreed upon continuation of the approach as an operational practice.

With regard to RQ5.2 (*To what extent does FSM-pricing help to improve transparency of project proposals?*) a noteworthy finding was that a large majority (84%) of the respondents of the survey experienced that transparency of project proposals is improved during the FSM-pricing pilot. We observed that the majority of discussions moved from effort (and price) estimate to waiver requests and getting requirements ready for FPA. Noteworthy is that FPA seems to have a positive effect on requirements management.

Looking at RQ5.3 (*To what extent does FSM-pricing help to create cost and time improvements?*) quantitative analysis of the performance of the *BelTel* projects taught us that Cost per FP improved during the study, where Duration per FP is not improving over time: this even shows a deterioration. This deterioration however seems to be caused by the fact that average project size gets smaller during the study while average project durations improve notably over time: average project duration in 2014 was even better than that of peer groups in industry.

5.7.1. Evaluation of Validity

Construct validity

With regard to the degree to which a test measures what it claims to be measuring a remark is in place on FPA. We used functional documentation as a source for FPA; a consequence is that low quality documentation could have led to low quality FPAs, however, we thoroughly reviewed all sets on completeness and correctness. Two (2) out of four (4) FPA specialists were certified; yet, all involved FPA specialists were highly trained and experienced FP-counters.

With regard to quality of data we argue that all project data was reviewed by the applicable *BelTel* project manager, all data on project cost was reviewed by the financial controller of *BelTel*, all project data was presented to and discussed with *BelTel* management.

Internal validity

We warranted the extent to which a causal conclusion is based on our study, by normalizing all project data with the functional size in FPs. In this way we were able to objectively compare performances of all projects in order to minimize systematic error. Based on the number of software projects, the diversity of projects and business domains within *BelTel*, and the fact that we measured and analyzed software project portfolios as a whole in an empirical way we argue that the effect of outliers is limited and that the risk on bias is mitigated responsibly.

External validity

Whether the study results can be generalized to settings outside the study, we argue that due to the limited scope of the performed case study (one sourcing company and one main supplier) it is too early to generalize the above mentioned findings to other companies and suppliers of software projects.

5.7.2. Relation to Existing Evidence

From our analysis of related work, it is clear that pricing in itself is a topic that has received little attention from the research community. Yet pricing is a topic of great practical value, which strongly affects the outcome (success or failure) of a software development project. The many budget overruns reported for such projects, may very well be more attributable to inadequate pricing than to poor project execution.

5.7.3. Impact/Implications

Our research shows that an evidence-based approach, in which historical data on key performance indicators are used in combination with a simple (power) regression, can lead to prices that are satisfactory to both suppliers and commissioning parties. It emphasizes a holistic approach, in which pricing is considered for the full IT portfolio of an organization, in combination with a supplier in an outsourcing relation. A major prerequisite for this approach is the availability of historical project data. This implies that the approach is only applicable to organizations willing and capable to aim for a long term solution.

The need for historical project data is likely also one of the causes why pricing has received limited attention in the research community; few researchers have access to such data. A way out of this dilemma may be opening up performance data for government-funded projects, making them available for researchers. Besides bringing new research insights, this might also help governments to reach more adequate prices for their IT projects.

5.7.4. Limitations

The reader should consider several limitations when interpreting our results. First, the survey has limited generalizability due to the limitation of respondents to 25 stakeholders. Determination of survey topics was done by members of both measurement teams, limited by the length of the survey (10-minutes). Further, the results of the ratings within the survey have to be looked upon with low significance in mind. We did not ask respondents to connect their open ended text data with the answers given in the rating part of the survey.

Second, we conducted the study only within *BelTel* and *IndSup-A*, so the results may not generalize elsewhere. Since we did not find any other study on a comparable single, statistical pricing approach, we cannot predict what the outcome of our method will be in other companies.

Third, our study focused on transparency of proposals and cost and duration improvement. The respondents might have been influenced by this focus and emphasize these aspects in their answers.

5.8. Conclusions and Future Work

The key contributions of this chapter are:

RQ5.1: We demonstrate that FSM-pricing is successfully used in practice of *BelTel* and *IndSup-A*, as a statistical, evidence-based pricing approach for software project proposals.

RQ5.2: We show that using FSM-pricing as a single instrument, without intervention of expert judgment-based opinions, leads in *BelTel* and *IndSup-A* to an improved transparency of project proposals and satisfied stakeholders from both the customer and the supplier.

RQ5.3: We demonstrate that FSM-pricing does lead to cost improvement within *BelTel* and *IndSup-A*. Cost per FP shows to be in line with external peer groups. Duration per FP on the contrary is too high when benchmarked externally and shows a deteriorating trend, probably caused by the fact that average project size gets smaller over time.

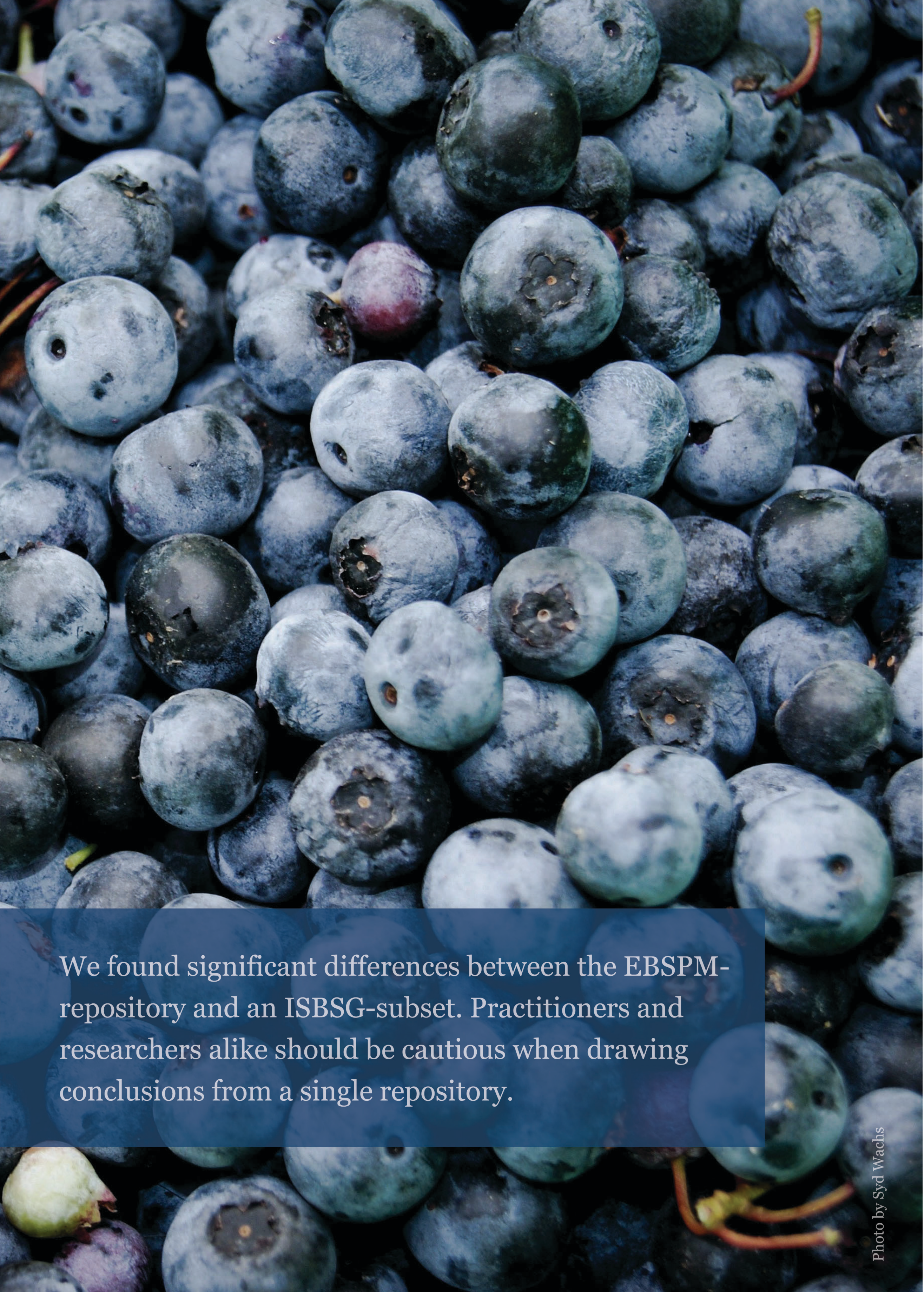
5.8.1. Future Work

The research presented opens up a number of avenues for further research. From a benchmarking perspective, our current approach distinguishes between data-warehousing and CRM/Billing projects. Further research is needed to come up with general guidelines on how to group projects into sufficiently cohesive units to permit adequate pricing. Another concern that arose from our case study is dealing with non-functional requirements such as security or infrastructure.

Delivery of smaller software projects in equal project durations seems to result in a lower Duration per FP; however, its needs to be researched whether the amount of value delivered by a project influences such performance perception. With regard to including non-functional requirements it might be interesting to perform future research on possibilities to use IFPUG SNAP (Software Non-functional Assessment Process) besides FPA. Approaches like COCOMO 2 introduce factors to compensate for such project characteristics, but whether this works well in combination with the purely statistical approach investigated in the present chapter calls for additional research.

5.9. Acknowledgments

We thank both *BelTel* and *IndSup-A* for their generosity to agree on using project and survey data for our study. Furthermore we thank Philippe Kruchten, Frank Vogelezang, Kim Herzig for their valuable feedback.



We found significant differences between the EBSPM-repository and an ISBSG-subset. Practitioners and researchers alike should be cautious when drawing conclusions from a single repository.

Photo by Syd Wachs

6. Effort versus Cost in Software Repositories

C*ontext:* The research literature on software development projects usually assumes that effort is a good proxy for cost. Practice, however, suggests that there are circumstances in which costs and effort should be distinguished. *Objectives:* We determine similarities and differences between size, effort, cost, duration, and number of defects of software projects. *Method:* We compare two established repositories (ISBSG and EBSPM) comprising almost 700 projects from industry. *Results:* We demonstrate a (log)-linear relation between cost on the one hand, and size, duration and number of defects on the other. This justifies conducting linear regression for cost. We establish that ISBSG is substantially different from EBSPM, in terms of cost (cheaper) and duration (faster), and the relation between cost and effort. We show that while in ISBSG effort is the most important cost factor, this is not the case in other repositories, such as EBSPM in which size is the dominant factor. *Conclusion:* Practitioners and researchers alike should be cautious when drawing conclusions from a single repository.

This chapter was published as *Effort versus Cost in Software Development: A Comparison of Two Industrial Data Sets* in the ACM Proceedings of the 21th International Conference on Evaluation and Assessment in Software Engineering (EASE 2017) (Huijgens, van Deursen, Minku, & Lokan, 2017c).

6.1. Introduction

A good understanding of the cost of software development, that is *the real cost that companies pay* for their software development activities, is important for business to make better decisions (Šmite, Calefato, & Wohlin,

2015) (Petersen, 2011). Two issues arise with regard to cost of software projects.

First, cost and effort are often looked upon as equivalent. At the best effort is assumed to be a good proxy for cost, where the emphasis seems to be more on effort, and less on cost. Frequently studies are found that claim to be about cost estimation, yet the study itself analyzes effort as the main subject, e.g. (Radliński, 2011) (Jeffery et al., 2000) (Pendharkar et al., 2009) (Czarnacka et al., 2009). To judge the evidence of cost-savings in global software engineering, (Šmite et al., 2015) reviewed more than five hundred articles on global software engineering, and found that only fourteen articles presented evidence of cost-savings (Šmite et al., 2015).

Second, many benchmarks are available for software projects. Jones (2011) mentions no-less than twenty-five sources of software benchmarks. (Menzies & Zimmermann, 2013) inventoried thirteen repositories of software engineering data. Yet, cost data are missing in most of them. One might expect that the idea of cost as an important factor for evidence-based steering on software engineering activities, combined with the availability of many sources for benchmarking, would lead to substantial knowledge about efficiency in terms of cost.

Yet, in practice this is not the case. To illustrate this, a recent study of (Bala & Abran, 2016) on how to deal with missing data in the repository that is maintained by the International Software Benchmark Standards Group (ISBSG, 2014), does not mention cost once, while a large part of the study is about effort related issues.

A systematic review of software development cost estimation studies by (Jørgensen & Shepperd, 2007) states that the "main cost driver in software development projects is typically the effort and we, in line with the majority of other researchers in this field, use the terms cost and effort interchangeably in this paper." Existing literature assumes that cost and effort are the 'same thing'. However, from interactions with industry, we believe the relation between both metrics is not that simple. The collection of these metrics needs to be done in different ways, each with their own difficulties. Furthermore, both metrics may be affected by different variables, such as country, inflation, and commercial aspects. While cost data might be reliable from an accounting viewpoint, they might include different actual data across projects phases. In order to emphasize the importance of cost, especially top-level decision makers highly value cost transparency, as we found in a study on pricing of software projects (Huijgens et al., 2015c). To explore the differences and commonalities between cost and effort, we use two large software repositories; the EBSPM repository (Huijgens, 2017a) and the ISBSG repository (ISBSG, 2014), with the objective of gaining insights into the usefulness of historic effort and cost data for benchmarking and cost estimation purposes. This is one of the first studies in which EBSPM is used in comparison with other benchmark datasets. We perform our analysis as an exploratory study based on two data sets.

Therefore, our datasets and results may not generalize. Our objectives are to (1) determine the similarities and differences between cost and effort, (2) determine whether these make cost modelling a substantially different problem from effort modelling, and (3) discuss potential reasons for the differences. We address the following research question:

RQ: How do the EBSPM-repository and the ISBSG-repository compare with regard to the size, effort, cost, duration, and number of defects of software projects?

When building the EBSPM-repository we experienced that effort and cost are not the same, mainly due to complex interactions with industry. A software company's project portfolio is usually built from differently organized cost structures. Simple structures like effort times hourly tariff are mixed with many other factors such as upfront agreed fixed price activities, delivery strategies such as agile (Scrum) where teams are budgeted for a period of one year or longer, sourcing strategies with globally distributed teams, and many more. Due to this the theory that cost can simply be calculated out of effort based on hourly rates is spurious, and does not hold for many software projects in industry.

We experienced in practice, that many decision makers in industry use cost as a major indicator for their decisions, and not effort as such. The interactions mentioned above make us believe that effort is not a simple proxy for cost, and that both metrics should be looked upon in research as an autonomous subject.

This chapter is structured as follows. In Section 6.2 we describe our research approach. Section 6.3 is about the results of our research. In Section

6.4 we discuss the outcomes and implications and threats to validity. In Section 6.5 we link these with related work. Finally, in Section 6.6 we describe conclusions.

6.2. Research Approach

For this exploratory study we apply a qualitative and a quantitative approach. We discuss differences between cost and effort, based on our previous interactions with industry, and we analyze correlations between cost and effort based on a subset of the ISBSG-repository. We create cost models based on two data sets; a subset of the ISBSG-repository and the EBSPM-repository, and we investigate whether typical results, such as influence of size and differences between data sets, obtained by the literature on software effort estimation also hold for cost.

We test whether both repositories are different from each other and in what degree they are fit for use to build a prediction model for effort and for cost. In particular we examine what independent variables are relevant for predicting cost. In both cases we look for the best fit; meaning that for each dataset different prediction models can apply. Furthermore, we evaluate the performance of both repositories from the perspective of a full software portfolio, by analyzing specific samples of software project data against the content of the EBSPM-repository (Huijgens, 2017a). Finally, we look for causes that might explain our findings, by studying existing literature on the subject of effort and cost of software projects.

6.2.1. The EBSPM-repository

The EBSPM-repository is a collection of data from approximately five hundred finalized software projects. Data is collected by specialized measurement teams within three different companies (banking and telecom) in The Netherlands and Belgium. Data is available for projects of different business domains. The functional size of all projects is measured in function points by specialized function point analysts. We focus on size, effort, cost, duration, and defects, which are all present in this data set, although effort is only collected for a limited number of 22 out of 488 projects. All software projects have been performed and measured in the period of nine years from 2008 to 2016. An important feature of the EBSPM-repository is that it contains data

Table 6.1. Descriptive Statistics of the EBSPM Project Data.

n = 488	Size (FPs)	Cost (Euros)	Effort (Hours)	Duration (Months)	Nr. of Defects
Minimum	4.00	329	31	0.90	0.00
First Quartile	38.25	71684	99	5.39	6.75
Median	115.50	293166	807	8.41	20.50
Mean	216.07	637935	3202	8.96	70.09
Third Quartile	248.75	740559	2391	11.09	55.25
Maximum	4600.00	7523527	22096	26.84	1586.00
Skewness	6.26	3.68	466 NAs	0.96	222 NAs

NAs indicate fields with no data available for effort and number of defects; due to this skewness could not be calculated. We emphasize that due to 466 NAs with regard to Effort only 22 projects are included in the calculations on Effort.

of a company's software portfolio as a whole, representing a variety of projects, business domains, delivery approaches, and sourcing strategies.

Where other repositories – like ISBSG – focus on projects as such, EBSPM focus on portfolios instead. This enables us to analyze *good practice* versus *bad practice* projects from a portfolio point of view (Huijgens et al., 2014c). In order to reduce effects of inflation we calculate all Euro values in the EBSPM-repository that we used for our study to the 2015 value, based on the calculation tables of the International Institute of Social History[2]. Table 6.1 shows descriptive statistics of the EBSPM-repository. The repository and the accompanying tool are described in more detail in a separate tool description (Huijgens, 2016a).

6.2.2. The ISBSG-repository

ISBSG is a repository collected by the International Software Benchmark Standards Group (ISBSG) (ISBSG, 2014), that is licensed to software companies that wish to use their tools for estimation and benchmarking purposes (for researchers ISBSG data is available free of charge). For the purpose of this study we use ISBSG version D&E Corporate Release 17 April 2013 (ISBSG, 2014). The full ISBSG-repository consists of 6010 projects; we select those projects for which there is data on cost, size, and duration:

[2] http://www.iisg.nl/hpw/calculate.php

Table 6.2. Descriptive statistics of the ISBSG project data.

n = 172	Size (FPs)	Cost (Euros)	Effort (Hours)	Duration (Months)	Nr. of Defects
Minimum	11.00	1627	183	1.00	0.00
First Quartile	40.50	23873	497	4.00	7,00
Median	125.00	68974	1445	10.00	13.00
Mean	307.40	180813	3890	11.31	98.85
Third Quartile	296.00	215861	3760	17.60	25.00
Maximum	10571.00	1915823	70035	54.00	2395.00
Skewness	9.90	2.96	5.80	1.02	(79 NAs)

1. We exclude all projects with no size recorded (*Functional Size* is empty).
2. We solely include projects that are counted in function points (values “IFPUG 4+”, or “NESMA” in *Count Approach*).
3. We exclude all other projects. We exclude projects with no cost recorded (*Total Project Cost* is empty).
4. We exclude projects with no project duration recorded (*project elapsed time* is empty).
5. Finally, we exclude all projects that were executed before the year 2000 (*year of project*), in order to limit the subset of projects to periods close to those in the EBSPM research repository.

After filtering, a subset of 172 projects is available for further comparison of effort and cost in the EBSPM-repository. In order to normalize all project cost in the subset, we convert the data in *Project Cost* to Euros based on historical exchange rates as denoted by trading company Oanda[3]. To do so we select the 1st of January of the applicable *Year of Project* in ISBSG as begin-date and the 31st of December of the *Year of Project* from ISBSG as end-date. Alike the EBSPM-repository we finally calculate all Euro values to the 2015 value, in order to reduce any effects of inflation.

Table 6.2 gives an overview of descriptive statistics of the ISBSG-subset that we used in this study. An comprehensive overview of the ISBSG dataset

[3] http://www.oanda.com/currency/average.

that we used, including the calculated cost data, is to be found in a technical report (Huijgens et al., 2016c).

6.2.3. Analysis Procedure

We perform a series of statistical tests to examine whether both repositories are significantly different. We perform Wilcoxon rank sum tests to compare differences between size, cost, effort, duration, and number of defects, and differences per size. To check for normality, we perform Shapiro tests and analyze histograms. To examine the prediction power of both datasets we use linear regression, stepwise linear regression, and CART trees, the latter as suggested in Nisbet et al. (2009). For linear regression, we also eliminate influential observations based on Cook's distance. Besides that, we compare both repositories by mapping the project data of the ISBSG subset on the *cost duration matrix* in the EBSPM-tool (Huijgens, 2016a), analyzing the following performance indicators:

1. *Cost per function point* (FP): the weighted average cost (in Euros) per FP, where size (FP) is the weighting factor, instead of number of projects.
2. Duration *per FP*: the weighted average duration (in calendar days) that it took to deliver a FP.
3. *Number of defects per FP*: the weighted average defects (from system integration test to technical go live) per FP.

We compare the performance of projects in the EBSPM-repository as a whole with ISBSG data. Within the scope of this study we do not look at company-specific causes; we look at 650 projects, and assume that company specific causes are not significant for common trends, as indicated in previous research (Huijgens et al., 2014c). Company-specific factors do not necessarily give us homogeneous information about aspects like country or sourcing strategy, because companies may be spread through different countries, their projects may be heterogeneous in terms of sourcing strategy.

6.3. Results

In order to examine the fitness of both repositories to build a prediction model for effort and cost, we performed a series of statistical tests. A detailed overview of the results of these tests is included in a technical report

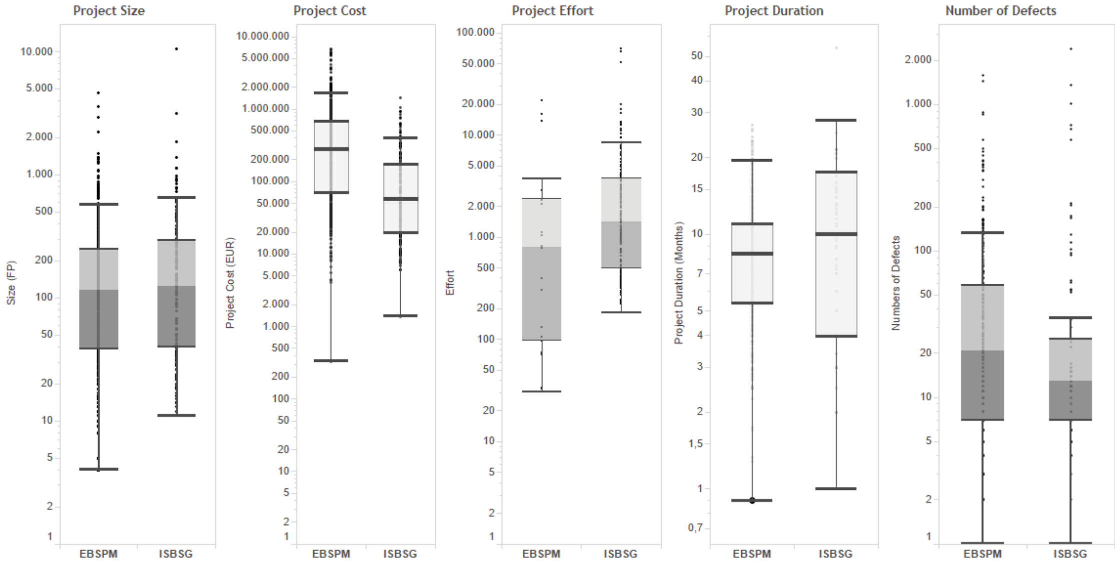


Figure 6.1 Boxplots showing the differences between the projects in the EBSPM-repository and the ISBSG-subset.

(Huijgens et al., 2016c). In this section we provide a summary of the most relevant outcomes.

In accordance with what is known from related work (Radliński, 2011) (ISBSG, Jones, & Reifer Consultants) (Huijgens & Vogelezang, 2016), we found that size, cost, and to a lesser extent duration, in both datasets were not normally distributed, indicated by a relatively high score for skewness (see Table 6.1 and Table 6.2). Boxplots of both repositories (see Figure 6.1) confirm this observation. We also checked for normality by performing Normality Shapiro tests. These result in violations for all numerical variables (all p-values were smaller than 2.4e-10; see the technical report (Huijgens et al., 2016c). To examine differences between both datasets we performed Wilcoxon ranked sum tests with Holm-Bonferroni corrections to compare overall differences, and differences per size (see Table 6.3). P-values indicate that *overall cost* and *cost per size* are significantly different, whereas the other metrics are not significantly different.

Table 6.3. Results from the Wilcoxon rank sum tests comparing EBSPM and ISBSG.

	Median	Median	W	p-value
Size	115.50	125.00	39063	0.1539
Cost	293166	68974	60446	0.0000*
Effort	807	1445	1369	0.0351
Duration	8.41	10.00	36768	0.0128
Number of Defects	20.50	13.00	13441	0.2592
Cost / Size	2684	602	72157	0.0000*
Effort / Size	NA	13.13	1668	0.3674
Duration / Size	2.09	1.62	46123	0.0650
Number of Defects / Size	0.18	0.31	11024	0.0974

The highlighted and with an asterix marked rows indicate statistically significant difference when applying Holm-Bonferroni corrections based on 7 comparisons, at the overall level of significance of 0.05; due to this correction Effort and Duration are not assessed significantly different.

6.3.1. Linear Regression

In order to further examine differences between both datasets, we use residuals versus fits and QQ plots to examine the points with values that are substantially larger than the rest. We perform several tests in order to examine which model fits best based on both datasets. As the data are not normal, we apply a log transformation, as recommended in the literature (Foss, Stensrud, Kitchenham, & Myrtveit, 2003).

A subset of the plots is shown in Figure 6.2; indicating the differences between the EBSPM-repository and the ISBSG-subset. The upper left plot (EBSPM) and the upper right plot (ISBSG) give an idea of whether the relationship between dependent and independent variables is linear. When the red lines are horizontal, the relationship is likely to be linear. The plots indicate that for both EBSPM and ISBSG the relationship deviate a bit from linearity, but these are small. Some violations to homoscedasticity (homogeneity of variance) are indicated, especially for ISBSG, as indicated by the plot at the bottom right.

Both QQ-plots at the bottom left (EBSPM) and bottom right (ISBSG) indicate fairly normal residuals for EBSPM. However, for ISBSG, the distribution seems less normal; indicating that linear regression is less adequate for ISBSG than for EBSPM. Application of backward stepwise linear regression

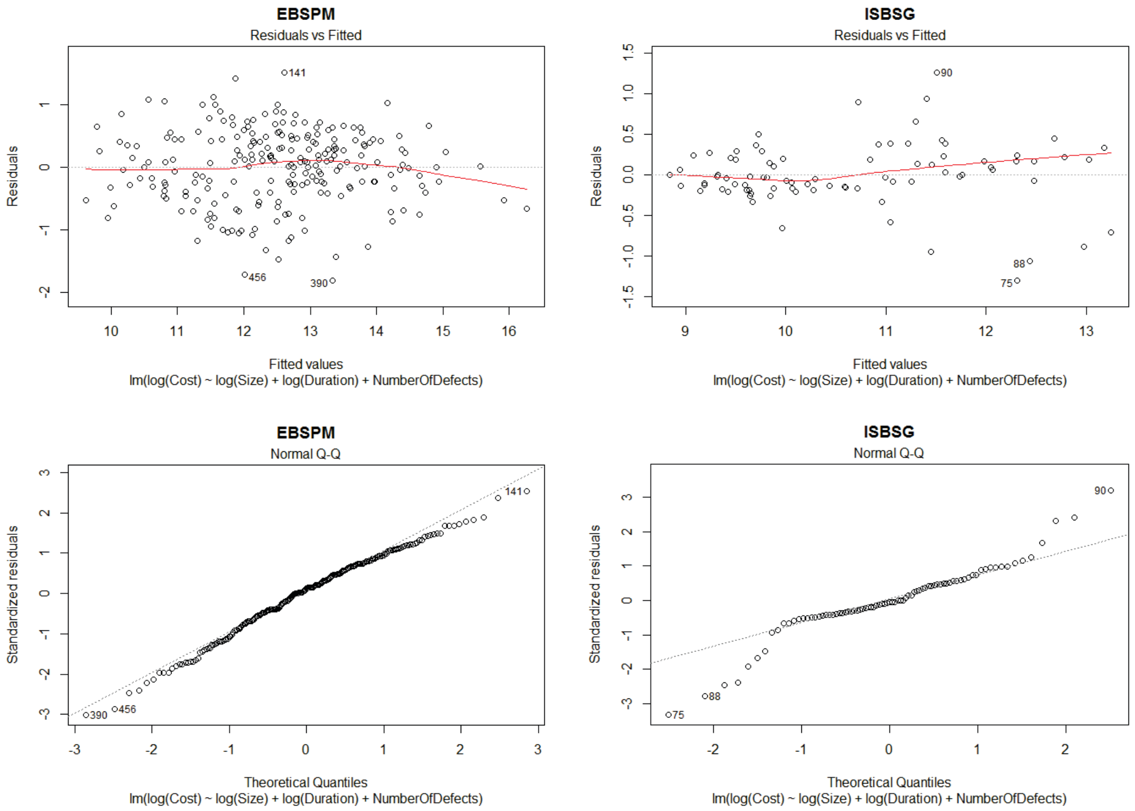


Figure 6.2. Comparison between Residuals and Fitted Values of the EBSPM-repository (top left) and the ISBSG-subset (top right), and QQ-plots of the EBSPM-repository (bottom left) and the ISBSG-subset (bottom right). The plots are generated on log(Size), log(Duration), and Number of Defects. Highly influential observations are removed from both datasets before the plots where generated. In all plots no log is applied for Number of Defects.

results in a similar fit as linear regression. We determined highly influential observations based on Cook's distance and the stepwise model (Bollen & Jackman, 1990); removing these in the stepwise model had no effect (see Table 6.4).

In Table 6.4 we show how much improvement in fit (multiple R-squared) we gain by using log, even though log is not always making things normal. We use linear (not stepwise) regression, so that we can know the estimate of the

Table 6.4. Improvements in fit (Multiple R-squared).

	EBSPM	ISBSG
Linear regression, without log	0.7663	0.1416
Linear regression, with log	0.7012	0.6815
Linear regression, no influential observations	0.8123	0.9029
Linear regression, with additional factors[1]	0.8839	0.9226

[1] For EBSPM the factors Organization, Business Domain, Development Approach, and Year Go Live were added. For ISBSG the factor Development Type was added. Effort was not included in any of the models above.

slope and result of the statistical tests of the hypothesis that the slope equals zero, with respect to all independent variables.

Note that effort is not included in the models; in terms of linear regression, excluding effort leads to a better R-squared (0.9029), than including effort (0.7159).

Our results indicate that for EBSPM adding effort does not help with linear regression. Fitting a linear model, with log transformation, where highly influential observations are removed before the model was generated, shows that in the EBSPM-repository *size*, *duration*, and *number of defects* are significantly related to *cost* (see Table 6.5). This confirms common knowledge from related work that *size* is a strong predictor of *cost*. In addition, it shows that for EBSPM *duration* and *number of defects* are strong predictors for *cost* too.

A similar test on the ISBSG-subset, shows that *size* and *effort* are relevant factors; however, *duration* and *number of defects* are not. A warning is in

Table 6.5. Results of fitting a linear model.

	EBSPM		ISBSG	
	Estimate	Star-rate	Estimate	Star-rate
(Intercept)	8.11367	***	5.53994	***
Log(Size)	0.58485	***	0.36232	**
Log(Duration)	0.35923	***	0.04552	
Log(Effort)	NA	NA	0.44362	**
Number of Defects	0.28517	***	0.09518	

Results of fitting a linear model, with log transformation in R. Highly influential observations are removed before the model was generated.

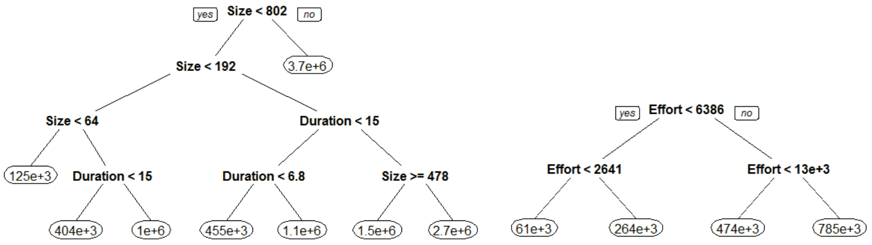


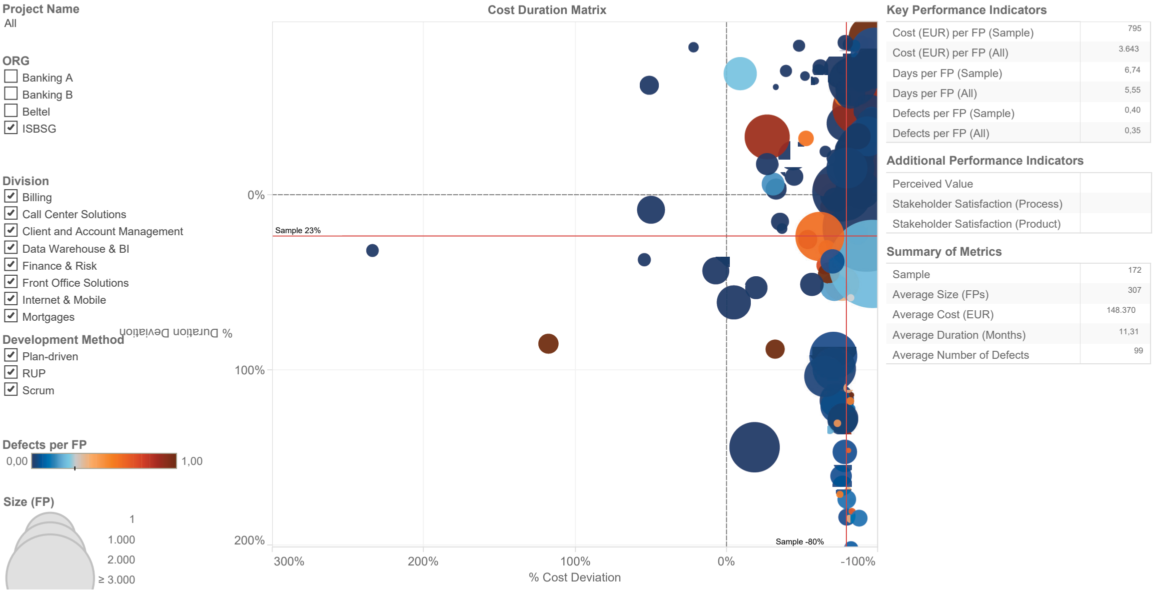
Figure 6.3. Regression Trees based on the EBSPM-repository (left) and the ISBSG-subset (right). In the EBSPM dataset not enough effort data was available to perform a regression tree with effort included.

place here; these results might be influenced by the fact that for EBSPM not enough effort data was available to fit a linear model (see the NAs in Table 6.5).

6.3.2. Regression Trees

Given the potential violations to the assumptions of linear regression (especially when using ISBSG), we created regression trees for both EBSPM and ISBSG, to see if they agree with our conclusions on linear regression. When examining the regression tree based on the EBSPM-repository (see Figure 6.3, left), we observe that *size* is the root node. This indicates that this is the most important attribute for predicting *cost*. *Duration* appears for the first time at the third level of the tree, indicating that *duration* is less important than *size*, but still relevant for predicting *cost*. *Number of defects* does not appear in the regression tree; it is considered as irrelevant for predicting *cost*. Examining the regression tree of the ISBSG-subset (see Figure 6.3, right), we find that *cost* and *effort* do have a strong relationship, as suggested in existing literature. *Duration* and *number of defects* are absent, indicating that these are not relevant for the purpose of estimating *cost* for ISBSG.

By aggregating the findings obtained by linear regression and regression trees, we can make the following observations. For EBSPM, both *size* and *duration* are very relevant, with *size* being more relevant than *duration*. This is reflected both in the linear regression and regression tree. *Number of defects* are less important and may or may not be relevant, given their smaller coefficient in linear regression and absence in the regression tree. For ISBSG, *size*



Project Name
All

ORG
☐ Banking A
☐ Banking B
☐ Beltel
☑ ISBSG

Division
☑ Billing
☑ Call Center Solutions
☑ Client and Account Management
☑ Data Warehouse & BI
☑ Finance & Risk
☑ Front Office Solutions
☑ Internet & Mobile
☑ Mortgages

Development Method
☑ Plan-driven
☑ RUP
☑ Scrum

Key Performance Indicators

Cost (EUR) per FP (Sample)	795
Cost (EUR) per FP (All)	3.643
Days per FP (Sample)	6,74
Days per FP (All)	5,55
Defects per FP (Sample)	0,40
Defects per FP (All)	0,35

Additional Performance Indicators

Perceived Value	
Stakeholder Satisfaction (Process)	
Stakeholder Satisfaction (Product)	

Summary of Metrics

Sample	172
Average Size (FPs)	307
Average Cost (EUR)	148.370
Average Duration (Months)	11,31
Average Number of Defects	99

Figure 6.4. The cost duration matrix in the EBSPM-tool showing a sample of 172 projects from the ISBSG repository, plotted against 488 projects from the EBSPM research repository. The analysis indicates that the ISBSG-projects performed on average 80% cheaper, but 23% slower than the projects in the EBSPM-repository. This figure was also used in a description of the EBSPM-tool (Huijgens, 2016).

and *effort* are relevant and *duration* and *number of defects* are not. This is reflected in the linear regression and partly in the regression tree.

6.3.3. Mapping of the ISBSG-subset on the EBSPM-tool

To visualize performance in terms of *size*, *cost*, *duration*, and *defects*, we mapped the subset of 172 ISBSG projects on the content of the EBSPM-repository, by using the EBSPM-tool (Huijgens, 2016) (see Figure 6.4). To analyze the overall performance of both repositories, we calculated three performance indicators (see Table 6.6).

Overall average *duration per FP* as measured in the EBSPM-repository (5.53) does not match ISBSG *duration per FP* (6.74), but differences are relatively small. The trendline for duration of the ISBSG-subset (the horizontal red line) is 23% below the EBSPM-trend (the horizontal dotted line), indicating that ISBSG projects on average took 23% longer to finalize than the

Table 6.6. Overview of Performance Indicators.

	EBSPM	ISBSG
Observations	488	172
Cost (Euros) per FP	3,630	795
Duration (Calendar Days) per FP	5.53	6.74
Number of Defects per FP	0.35	0.35

All performance indicators are calculated as weighted average, with size as weighting factor (instead of number of projects).

EBSPM ones. On quality no differences occur, as both datasets show a weighted average of 0.35 *number of defects per FP*. The median *number of defects* is statistically similar according to the Wilcoxon tests.

Yet, differences on cost are huge. Average Cost per FP in the EBSPM research repository is 3,630 Euros, while the ISBSG repository shows an average of 795 *cost per FP*. The ISBSG trendline in Figure 6.4 (the vertical red line) is as much as 80% to the right of the EBSPM-trend (the vertical dotted line), indicating that ISBSG projects were 80% cheaper in terms of *cost per FP* than EBSPM ones.

With regard to project size (in function points) we observe that on average ISBSG projects show a size of 307 FPs, where the EBSPM projects show an average size of 216 FPs. However, these differences in size do not explain the huge difference in cost in both datasets. This is confirmed by the Wilcoxon Ranked Sum Tests with Holm-Bonferroni corrections, which reveal no significant differences in *size* between EBSPM and ISBSG.

6.3.4. Key Findings

Looking at our research question:

RQ: How do the EBSPM-repository and the ISBSG-repository compare with regard to the size, effort, cost, duration, and number of defects of software projects?

we determine the following key findings in this study:

1. We demonstrate a (log)-linear relation between cost on the one hand, and size, duration and number of defects on the other (as illustrated in Figure 6.2). This justifies conducting linear regression for *cost*.

2. We establish that ISBSG is substantially different from, e.g., EBSPM, in terms of *cost* (cheaper) and *duration* (slower), and the relation between *cost* and *effort*. This implies that practitioners and researchers alike should be cautious when drawing conclusions from a single repository.
3. We show that while in ISBSG *effort* is the most important cost factor, this is not the case in other repositories, such as EBSPM in which *size* is the dominant factor.

6.4. Discussion

The main questions that arise from the analysis that we performed, is whether the large differences that we found in *cost*, and the fact that *duration* and *number of defects* are of influence to *cost* in the EBSPM-repository, and not in the ISBSG-subset, can be explained in any way?

Our analysis shows that the EBSPM-repository and the ISBSG-subset are significantly different with regard to cost of finalized projects. Looking at the significant differences in the EBSPM-repository between different companies, as shown in the boxplots in Figure 6.1, we tend to agree with the idea that building a dedicated repository of historic projects (a single-company model) helps companies to make better predictions of new projects and to improve benchmarking and analysis of ongoing and finalized projects. Also the idea of clustering fits with our experience that some business domains (e.g. data warehouse) show cost patterns that deviate from general ones, and therefore need to be looked upon in a specific way.

Based on the outcomes of our comparison, we argue that, effort and cost are interrelated, at least in the ISBSG-subset that we studied. However, when plotted over time both metrics seem to show a more complex relationship (see the technical report (Huijgens et al., 2016c) for a figure on development of project metrics over time). Besides the common idea that cost reflects effort times hourly tariff, many other factors play a role here: e.g. market issues, productivity changes over time, and sourcing strategies. However, no major cost per FP changes are to be found in ISBSG data when looked upon over time. What strikes in our study, however, is the remarkable difference in average weighted *cost per FP* between both studied repositories, as depicted in Table 6.6; in the ISBSG-subset development of one function point cost on average 795 euro, while in the EBSPM-repository 3630 euros are needed to do so.

The big differences that we found between project cost in both repositories could not be explained based on *size*, *duration* and *number of defects* of the software projects. Unfortunately, no relevant *effort* data was available in the EBSPM research repository, so we cannot conclude anything about that. In earlier research (Huijgens et al., 2014c) we found indications that might be of influence to cost of projects, such as software delivery strategies (agile and release-based, steady heartbeat, fixed teams) and specific business domains.

Unfortunately, the ISBSG descriptions do not tell us much about these aspects. A key factor in the comparison of both repositories might be different labor costs in different countries. EBSPM projects come from the Netherlands and Belgium, however a significant part of the projects is performed in cooperation with suppliers in other countries (e.g. India). ISBSG projects come from a wide range of countries, including some whose IT industry is based on labor costs being low. Since ISBSG does not normally release data about the country in which a project was performed, it is hard for users of ISBSG cost data to take this into account. However, when the total cost of the ISBSG projects in the subset that we used are divided by the total effort, this results in an average hourly rate of 43.01 Euro. A complicating factor here might be that in the ISBSG repository, total project effort is a mandatory field (Déry & Abran, 2005), where cost is not. In the EBSPM-repository; cost is mandatory, mainly due to the recurring difficulties that the measurement teams involved in EBSPM experienced in collecting reliable effort data in practice.

One major aspect that we assume to be key in this comparison, is that in the EBSPM-approach a software portfolio as a whole of each company is measured, e.g. the good and the bad projects. The ISBSG-approach focuses at single projects, and therefor might not reflect a company's portfolio performance in a holistic way.

6.4.1. Implications

Taking into account that collecting reliable effort in industry is difficult, and that we found substantial differences between both studied datasets, we point out that researchers and practitioners should take great care to understand the data they are using when making estimates. A remark with regard to using repositories for prediction purposes, is that not only the adequacy of different predictive models for each dataset varies, but also the most relevant independent variables. This is in line with previous research on effort estimation (Minku & Yao, 2013). Care must also be taken when using data from one repository to predict projects for companies not represented in the repository. As we show, projects from different repositories can be considerably different. In particular, we observe that the two repositories had projects with similar duration and number of defects (independent variables), but different costs: relevancy filtering and locality-based approaches, which have been achieving promising results for cross-company effort estimation (Turhan & Mendes, 2014) (Kocaguneli, Menzies, & Mendes, 2015), might not work well for predicting cost.

6.4.2. Threats to Validity

Our study focuses on differences between size, cost, duration, and number of defects in both repositories. All other factors, such as organization, business unit, primary programming language, development approach, or development type are not included in the analysis. We are aware of the fact that including different factors might influence the outcome of the analysis, yet we would like to argue in our favor that overall both repositories are a representative subset of any company's software project portfolio. Statistical tests showed no evidence that organization or business domain was of significant influence for cost prediction, where size, duration and numbers of defects actually were. As a remark we mention that related work shows that a relationship between cost and business domain is applicable in the EBSPM-repository (Huijgens et al., 2014c), and with effort in the ISBSG repository (Lokan, Wright, Hill, & Stringer, 2001).

Two important limitations might be of influence to our study. Only a small part of the software projects in the ISBSG-repository includes cost data; a subset of 172 out of 6010 projects (3%) was applicable for our analysis. On the contrary, only a small part of the EBSPM-repository holds effort data; 22 out of 488 projects (4.5%). We emphasize that our findings are not to be generalized without any restrictions to other software repositories. In order to assure the quality of Function Point counting in the ISBSG-subset; the subset only contains projects with an Unadjusted Function Point Rating 'A' (the unadjusted FP was assessed as being sound with nothing being identified that might affect its integrity) or 'B' (the unadjusted function point count appears sound, but integrity cannot be assured as a single figure was provided).

With regard to quality assurance of the EBSPM-repository: projects were measured by experienced, often certified measurement specialists. Project data was based on formal project administrations and reviewed by stakeholders (e.g. project managers, product owners, finance departments, project support). All projects were reviewed thoroughly by the first author of this study before they were included in the EBSPM-repository. We used the default parameters from the CART package in R to build regression trees.

Finally, we emphasize that we used – where possible – a relevant and extended subset of statistical tests to analyze both repositories. Our goal was to link evidence found from one test to confirming results from other tests too.

6.5. Related Work

6.5.1. Repositories for Benchmarking

Our findings are not all new. Related work confirms large differences between both within-company and cross-company repositories. Much research is performed on whether organizations should use cross-company datasets for estimation and benchmark purposes or whether they should collect their own historic data (Jeffery et al., 2001) (Briand et al., 2000) (Wieczorek & Ruhe, 2002) (Lokan & Mendes, 2006) (Minku et al., 2015) (Mendes et al., 2005). The outcomes of studies are mixed.

(Garre et al., 2005) emphasize that "in the case of large project databases with data coming from heterogeneous sources, a single mathematical model cannot properly capture the diverse nature of the projects". Apparently, other benchmark sources are important. Also the usually "large disparity of their instances" lead to problems (Garre et al., 2005). Many researchers have now a better awareness that single companies can themselves have heterogeneous projects. As a consequence, the community has started to question the usefulness of the distinction between the terms 'cross' and 'within' (Minku, 2016).

(Abran et al., 2002a) mention a lack of historical data in many companies, as a solution they propose a simulation using the ISBSG-repository. (Lokan et al., 2001) position ISBSG as a 'low cost initial determination of an organization's industry position and its comparative strengths and weaknesses'. (Fernández-Diego et al., 2014) inventoried the use of the ISBSG-repository by performing a systematic mapping review on 129 research papers. They found that in 70,5% of the studies prediction of effort is the main focus. In 55% of the papers ISBSG is used as the only support.

(Oligny et al., 2000) propose a duration prediction model that is based on ISBSG data, that can deliver a 'first order' estimate to project managers. (Lokan et al., 2001) state that "ISBSG does not claim that the repository represents the whole industry; rather, it believes that the repository only represents the best software companies" (Lokan et al., 2001). (Cheikhi & Abran, 2013) mention the ISBSG repository and Promise as the two ongoing repositories of software projects in the SE community, both lacking structured documentation, which hinders researchers to identify the datasets that are suitable for their purposes. (Déry & Abran, 2005) mention that "a key challenge in data analysis using the ISBSG repository (...) is to assess the consistency of the effort data collected".

6.5.2. Effort versus Cost

Effort and cost are in many studies used as equivalent; usually cost is mentioned, where actually effort is meant. An example is the definition by (Buglione & Ebert, 2011): "An estimate is a quantitative assessment of a future endeavor's likely cost or outcome". (Petersen, 2011) argues that both are an important decision criterion for companies. A recent study on productivity in agile software development (Shah et al., 2015) mentions cost as one of nine highly-related productivity dimensions, although the original study by (Melo et al., 2011) of two agile teams revealed that most team members did not share the same understanding of productivity. As a consequence of the blurred distinction between effort and cost, the latter is for a major part missing in studies on project performance. (Radliński, 2011) gives an overview of variables used in analysis within the ISBSG repository, yet cost is not mentioned. When cost actually is mentioned, it often is as equivalent to or a derivative of effort, see for example (Pendharkar & Rodger, 2009), (Czarnacka-Chrobot, 2009), and (Jeffery et al., 2000). (Deng & MacDonell, 2008) propose an approach based on justified normalization of functional size, to challenge questions among researchers about the quality and completeness in the ISBSG-repository.

Both the EBSPM-repository and the ISBSG-subset use functional size (Function Points) as metric to normalize the projects in their repositories.

Our study shows that in both repositories Size is strongly significant for Cost of software projects, an effect well known from other related work (Gencel & Demirors, 2008). This emphasizes the importance of including functional size in any form in a software project repository. It might even be an important next step to automate the counting of functional size, although recent research indicated that stakeholders on functional size measurement do not see a direct need for this (Huijgens et al., 2015b). However, we assume that this opinion might be slightly tainted by self-interest.

Although we used regression analysis for our study a warning is in place: (Jørgensen & Kitchenham, 2012) argue that violations of essential regression model assumptions in research studies to a large extent may explain disagreement among researchers on economy of scale effects or diseconomy of scale effects with regard to size and effort. Randomized controlled experiments with fixed software sizes and random allocation of development of software of different sizes, and the use of more in-depth of analyses of software projects might help here (Jørgensen & Kitchenham, 2012). (Radliński, 2011) examined how various project factors in ISBSG are related with the number of defects, finding that there are few factors significantly influencing this aspect of software quality. Such results might suggest that software quality depends rather on a wider set of factors (Radliński, 2011). This confirms what we found with regard to number of defects in the ISBSG-subset. Finally, no scientific studies are published that examine the link between effort and cost of software projects based on real industry data.

6.6. Conclusions

We compared two industrial yet publicly available software project repositories, the EBSPM-repository and a subset of the ISBSG-repository, in order to analyze differences with regard to Cost, Size, Duration, and Number of Defects. We determined suitability of some key variables (Size, Duration and Number of Defects) of both data sets for the purpose of cost prediction. We identified three key findings:

1. We demonstrate a (log)-linear relation between *cost* on the one hand, and *size*, *duration* and *number of defects* on the other. This justifies conducting linear regression for *cost*.

2. We establish that ISBSG is substantially different from, e.g., EBSPM, in terms of *cost* (cheaper) and *duration* (faster), and the relation between *cost* and *effort*. This implies that practitioners and researchers alike should be cautious when drawing conclusions from a single repository.
3. We show that while in ISBSG effort is the most important *cost* factor, this is not the case in other repositories, such as EBSPM in which *size* is the dominant factor.

We showed that *effort* and *cost* of software projects in the ISBSG-subset are interrelated, although results might be influenced by definitional issues with regard to *cost* and because we examined a subset of only 3% of the ISBSG-repository. We argue that, supported by the importance of both *effort* and *cost* data for decision makers in industry, *effort* and *cost* should be treated as different metrics in research.

6.7. Acknowledgments

Our thanks to Tableau for allowing us to use their business intelligence solution to build the EBSPM-tool and ISBSG for allowing us to use their repository for research purposes.

We found that a focus on shortening overall project duration, and improving communication and team collaboration on intermediate progress is likely to have a positive impact on stakeholder satisfaction and perceived value.

Photo by Todd Diemer

7. Stakeholder Satisfaction and Perceived Value

Context: In this chapter we present a multiple case study on the insights of software organizations into stakeholder satisfaction and (perceived) value of their software projects. Our study is based on the notion that quantifying and qualifying *project size*, *cost*, *duration*, *defects*, and *estimation accuracy* needs to be done in relation with *stakeholder satisfaction* and *perceived value*. *Objectives:* We contrast project metrics such as *cost*, *duration*, *number of defects* and *estimation accuracy* with *stakeholder satisfaction* and *perceived value*. *Method:* In order to find out whether our approach is practically feasible in an industrial setting, we performed two case studies; one in a Belgian telecom company and the other in a Dutch software company. *Results:* In this study we evaluate 22 software projects that were delivered during one release in the Belgian telecom company, and 4 additional large software releases (representing an extension of 174% in project size) that were delivered in a Dutch software company. Eighty-three (83) key stakeholders of two companies provide *stakeholder satisfaction* and *perceived value* measurements in 133 completed surveys. *Conclusions:* We conclude that a focus on shortening overall project duration, and improving communication and team collaboration on intermediate progress is likely to have a positive impact on *stakeholder satisfaction* and *perceived value*. Our study does not provide any evidence that steering on costs helped to improve these.

This chapter was published as *The Effects of Perceived Value and Stakeholder Satisfaction on Software Project Impact* in Information and Software Technology (Elsevier 2017) (Huijgens, van Deursen, & van Solingen, 2017d).

7.1. Introduction

An often cited result of the Standish CHAOS research (International Standish Group, 1994) is that 70% of all software projects are problematic. Standish defines these as so-called ‘challenged projects’, meaning they were not delivered on time, within cost, and with all specified functionality (Jørgensen & Moløkken-Østvold, 2006).

This is in a certain way along the lines of what we found when studying a series of 22 finalized software projects in a Belgian telecom company. We found that the average cost overrun was 28% (ranging from -41% to 248%), and that the average duration overrun was 70% (ranging from 9% to 168%). There was only one single project that performed within a 10% cost and duration overrun boundary. As such, these projects were challenged if we adopt the way Standish defines success and failure; being the extent in which a project conforms to its original plan.

However, did all the other 21 projects fail? Is it fair to say that a project with cost overrun is a failure? Is it reasonable to say that a project that performed completely according to plan, but delivered software that no one uses, is a success?

7.1.1. Problem Statement

Supported by many critical reviews of the Standish criteria (Jørgensen & Moløkken-Østvold, 2006) (Glass, 2006) (Eveleens & Verhoef, 2010), we define success and failure in this paper from a different angle, trying to include the balance between *value* and *cost* into the equation. In previous research we defined success and failure in terms of cost, duration and number of defects of a software project (Huijgens et al., 2014c) (Huijgens et al., 2015c) (Huijgens et al., 2016c). Looking at the outcomes of this we consider that a project that is late and over budget – and thus in terms of our study *bad practice*, or in other words unsuccessful – yet returns high value according to its stakeholders, may still be called successful, because of the fact that it delivers high value.

By analyzing project metrics such as *cost*, *duration*, *defects*, and *size* of the projects in connection with *stakeholder satisfaction*, *perceived value* and *quality of estimations*, we show that stakeholders define success and failure of a project different from solely measuring cost and duration overrun. Especially in domains where value is more important than predictability, e.g. agile ways of working, a limited view on conformance to planning, seems illogical. Due to the fact that measuring the real – delivered – value of software deliveries is difficult, we focus in this chapter specifically on *perceived value*. The underlying idea is that, since finding evidence in the bottom-line financial administration is hard, if not impossible, the best we can do is involve stakeholders for a qualitative indication of value. However, as this is strongly dependent on the individual and the contextual setting (what is valuable in one setting might not be valuable in another, or what one stakeholder considers to be of no value can be of high value to another stakeholder), we use the term *perceived value*. We understand that this is a way to measure value that is limited in its external and construct validity. However, this approach may help in finding early ways of indicating value (Gilb & Finzi, 1988).

In this chapter, we analyze a set of projects conducted at a Belgian telecom company (referred to in this thesis as *BelTel*) and a Dutch software company (referred at in this thesis as *DutchCo*) that provides billing software products and services (also largely to the telecom domain). We propose the following research question:

RQ: How do stakeholder satisfaction and perceived value relate to cost, duration, defects, size and estimation accuracy of software projects?

In answering this question, we make the following contributions:

1. We propose a light-weight value measurement technique based on post-release interviews.
2. We provide data on 26 industrial projects for which 83 key stakeholders provide *stakeholder satisfaction* and *perceived value* measurements in 133 completed surveys.
3. We contrast the resulting *perceived value* and *stakeholder satisfaction* statements with collected data on *costs*, *duration*, *defects*, *size* and *estimation accuracy* and look for links between them.

This chapter is an extended journal version of an earlier published paper at the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE 2016) (Huijgens et al., 2016c). Compared to the original paper the new contributions can be summarized as follows:

- We replicated the research performed in our original study in another company: *DutchCo*, a Dutch software company, specialized in delivering billing solutions to European telecom operators.
- Within *DutchCo*, we examined four (4) large software releases, representing an extension of 174% in project size. We collected detailed *size*, *cost*, *duration*, and *defects* data from all releases. We performed electronic surveys on *stakeholder satisfaction* and *perceived value* among thirty (30) stakeholders within the *DutchCo* organization and *IndSup-B*, its provider of India-based development teams.

The remainder of this chapter is structured as follows. In Section 7.2 related work and the background of the model that we use for analysis purposes are described. Section 7.3 outlines the research design. The results of the study are described in Section 7.4. We discuss the results in Section 7.5, and finally, in Section 7.6 we make conclusions and outline future work.

7.2. Background and Related Work

Many studies include critical reviews of the Standish CHAOS Report (Jørgensen & Moløkken-Østvold, 2006) (Glass, 2006) (Eveleens & Verhoef, 2010) (Moløkken & Jørgensen, 2003) (El Emam & Günes Koru, 2008) (Dybå et al., 2005) (Glass, 2005) (Sutherland et al., 2007). The Standish Group reported in their 1994 CHAOS report that the average cost overrun of software projects was as high as 189%. (Jørgensen & Moløkken-Østvold, 2006) conclude that this figure is probably much too high to represent typical software projects in the 1990s and that a continued use of that figure as a reference point for estimation accuracy may lead to poor decision-making and hinder progress in estimation practices (Jørgensen & Moløkken-Østvold, 2006). (Glass, 2006) states that objective research study findings do not, in general, support those Standish conclusions.

Where in our research we measure value as perceived by stakeholders on four business related subjects, many different measures are used to identify value, and a clear and uniform definition is no question yet. (Pekki, 2016) defines stakeholder value as the “usefulness of offering SPI to its key beneficiaries, so they are fully involved into SPI activities which increases the success of those activities”. (Beck, 2000) indicates that value is about money and time, by saying we “need to make our software economically more valuable by spending money more slowly, earning revenue more quickly and increasing the probably productive lifespan of our project”. (Dingsøyr & Lassenius, 2016) answer the question “What is value”? by saying that “the improvement trends are not specific on how they define value”. They come up with the argument that, ”proponents of agile development would argue that a development team needs to learn what external stakeholders value during a development project”. In a way this matches our idea that besides internal stakeholders, especially external stakeholders should be involved in the value discussion.

(Atkinson, 1999) argues that besides time, cost and quality, often referred at as *the iron triangle*, also stakeholder benefits should be taken into the equation. Besides that, he mentions the effect that quality is “an emergent property of people’s different attitudes and beliefs, which often change over the development life-cycle of a project”.

Estimate the value of software is probably as challenging as predicting the cost of software (Shepperd, 2014). (Strand & Karlsen, 2014) suggested to estimate value in the form of “benefit points”, as a kind of equivalent to story points. (Cheng et al., 2016) describe an architecture-based approach to discover value of software engineering by using big data techniques. Although quite some research has been performed in the area of value estimation (Boehm, 2003) (Biffl et al., 2006) (Faulk et al., 2000), and success criteria for software projects (Agarwal & Rathod, 2006) (Bryde, 2005), most of these approaches seem poorly adopted in industrial software project management settings. A good sign however, is that an increased focus on value in improvement is seen in software development, mainly driven by agile development approaches (Dingsøyr & Lassenius, 2016).

(Jørgensen, 2016) performed a survey among software professionals in Norway on the characteristics of projects with success in delivering client benefits. He mentions that a focus on client benefits as a success criterion is particularly important, because only weak correlations are found on other dimensions, such as “being on time” and “being on budget”. Besides that, he mentions that the traditional success factor “having the specified functionality” may even be in conflict with success in delivering client benefits.

7.3. Research Design

The goal of this study is to understand the underlying reasons of stakeholder satisfaction and value of software projects. To achieve this, we contrast project metrics such as cost, duration, number of defects and estimation accuracy with *stakeholder satisfaction* and *perceived value*. We argue this will help to better understand the backgrounds of software projects as a guide for building future software portfolios.

As explained in the introduction, the Standish criteria (International Standish Group, 1994) states that success and failure are related to the quality of project estimates. In order to explore alternatives, we test for association between paired samples, using Pearson's product moment correlation coefficient and resulting p-values in case our data is normally distributed or Spearman Rank Correlation when the data is not normally distributed. To mitigate the risk that we find coincidental correlations we perform an exploratory study that confronts correlated metrics with findings from qualitative results from analysis of the free format text from the surveys.

We performed a multiple case study in two different companies: *BelTel*, a Belgian telecom company, and *DutchCo*, a Dutch software company that delivers billing solutions to European telecom operators. In the following two Subsections we describe the industrial context of how both companies are included in our research.

7.3.1. *BelTel*

BelTel is a Belgian telecom company that can be characterized as a typical mid-sized information-intensive company with a mature software delivery organization that offers a mix of delivery approaches, ranging from plan-driven to agile (Scrum) (Schwaber & Sutherland, 2011). For the majority of its software development activities *BelTel* has a strategic, long-term contract with one large Indian supplier, referred to in this thesis as *IndSup-A*. Projects relate to different business domains (e.g. Internet, Mobile Apps, Data warehouse, Billing, Customer Relationship Management).

During the past three years, *BelTel* has adopted a metrics program to collect data on size, cost, duration, the number of defects, and the estimation accuracy of finalized software projects. This data has been used to analyze project performance at *BelTel*, to benchmark project performance, and to continuously improve the software delivery process within *BelTel*. In October 2015, *BelTel* changed its strategic focus from cost-based (steering on efficiency and operational excellence) to value maximization and shortening time-to-market. To facilitate this, *BelTel* has collected additional data, addressing business value and customer satisfaction.

In the present chapter, we compare these with the data on costs and duration that were also collected, in order to better understand the relationships between various project success indicators. Development projects at *BelTel* are conducted independently, yet are grouped for deployment into so-called releases. Once a project passes its system test it is promoted to a release, which typically contains multiple projects. Releases are further tested and deployed as a whole. Within *BelTel* eight subsequent releases are performed each year. In this chapter, we study data from 22 projects coming from four different releases.

7.3.2. *DutchCo*

DutchCo is a Netherland's-based software company that offers billing solutions to a large variety of European telecom companies. Within this market *DutchCo* is a European market leader.

Unlike *BelTel*, *DutchCo* does not structure its work into projects. All software development activities are organized into four large market releases each year. Driven by the desire of its customers to limit the number of deployments, *DutchCo* implements only four market releases a year. As a result, these four releases are usually quite large in size. Where *BelTel* thus implements eight releases a year, each of which consist of a large number of small and medium-sized projects, *DutchCo* performs only four large releases, which are composed of many small user stories.

To build and test its software, *DutchCo* makes use of several development teams in India (Šmite & van Solingen, 2016). These teams are supplied and supported by *IndSup-B*, a Dutch consultancy company, specializing in agile software delivery. Activities such as preparation of releases, design, quality assurance, and overall management are performed by members of an onsite, Netherland's-based team of *DutchCo* itself.

Based on the results of previous research within the organization, *DutchCo* pays considerable attention to communication between the different members of a development team. There is a virtual contact window that is constantly open to allow team members in different locations to contact colleagues, and substantial effort is put into reciprocal visits to the team sites.

All teams within *DutchCo* – including the development teams of *IndSup-B* in India – work according to the Scrum approach (Schwaber & Sutherland, 2011). An enterprise backlog and sprint backlogs are maintained in Jira, two-weekly sprints are performed, results are demonstrated to business stakeholders, and bi-weekly retrospectives are performed. As such the *DutchCo* market releases contain a combination of about 6 to 7 (bi-weekly) Sprint deliveries. As these Sprints do deliver working tested software, one could also call these releases. However, as these are only deployed in an acceptance test environment and not to the market, we use the term 'market release' for those four releases each year.

The *DutchCo* teams are organized in a component-based way. One database-team (DB) is based in The Netherlands. Two teams are based in India; one portal and asset management-team of nine people (POR and AM), and one reporting-team (AR) of also nine people.

Table 7.1 summarizes the release approaches of both companies. *BelTel* runs single projects that are combined eight times per year for user acceptance testing and deployment. In the *DutchCo* case no projects are to be found; user stories are combined in releases that are deployed every three months.

7.3.3. Challenges in Comparing both Companies

Looking at the large differences in the project size, staff count, budgets, geographic location of team and customer demands we recognized major challenges in comparing software projects performed in a telecom company with a software company. To remedy this, we used a tool that we designed to address this challenge (Huijgens, 2016a). In previous research we built a model, the so-called *cost duration matrix* (see Subsection 2.3.2 for an explanation), based on the consideration that *project size*, *project cost*, *project duration* and the *number of defects* detected during a software project are interrelated with each other (Huijgens et al., 2014c) (Huijgens et al., 2015c) (Huijgens et al., 2016c). The model takes a project's size, measured in function points (FPs) (IFPUG, 2009), as starting point and as a source for normalization that makes it possible to compare software projects with different settings. The model compares the actual costs normalized to a

Table 7.1. Summary of Release Approaches

	BELTEL (N = 4)	DUTCHCO (N = 1)
Frequency of Release	6 weeks (8 releases per year)	3 months (4 releases per year)
Scope of Release	Collection of projects from different Business Domains (a mix of Scrum and plan-driven)	Collection of User Stories performed by 4 Scrum-teams (of which 3 offshore in India)
Average Size of Release	444 Function Points	776 Function Points
Average Cost of Release	2,190 K Euro	512 K Euro

function point (in Euros per FP) and duration (in *days per FP*) for a project of this size to benchmarked data, taken from a set of 492 finalized software projects in the financial and telecom application domains. This is done using two power regressions conducted on the 492 projects, permitting the computation of the 'expected' cost and duration of a project of a given size (measured in function points) (Huijgens et al., 2014c) (Huijgens et al., 2015c).

7.3.4. Metrics

In this Subsection we describe and explain the major metrics that are collected and analyzed for the subject projects.

Project Metrics

Four project metrics are collected on each project that is subject of the case study: *project cost* (in Euros), *project duration* (in months), and the *number of defects* found during the project. *Project size* is measured in function points, according to the IFPUG industry standard (IFPUG, 2009). Based on this, we determine the *cost per function point*, *days per function point*, and *defects per function point*, using in each case the size in function points as weighting factor.

Estimation Quality Factor

The *estimation quality factor* (EQF) is a measure of the deviation of a forecast to the actual cost or duration. EQF is a forecasting metric that depicts the quality of forecasts made during a project. The measure was defined by (DeMarco, 1984). He defines EQF by:

$$EQF = \frac{Area\ under\ actual\ value}{Area\ between\ forecast\ and\ actual\ value}$$

We use the formalization proposed by (Eveleens & Verhoef, 2009). We reiterate and correct the definition given there. Let a be the actual value ($a >$ 0), t_a the time the actual is known and $e(t)$ the value of the forecast at time t ($0 \leq t \leq t_a$) in the project. Then, the EQF is represented by (Eveleens & Verhoef, 2009):

$$EQF = \frac{\int_0^{t_a} a\,\mathrm{d}t}{\int_0^{t_a} |a - e(t)|\mathrm{d}t}$$

$$= \frac{\int_0^{t_a} 1\,\mathrm{d}t}{\int_0^{t_a} |1 - e(t)/\,a|\mathrm{d}t}.$$

EQF allows us to quantify the quality of forecasts. A low EQF value means that the deviation of the forecasts to the actual cost or duration is large. EQF is measured for both cost and duration.

Cost Duration Index

The *cost duration index* is a measure of the relative position of a project within the *cost duration matrix* (see Figure 7.2). The index is represented as a number between zero and one hundred. In practice most projects score between 80 and 99. A high index corresponds to a good position in the *cost duration matrix* (best is top-right in the *good practice* quadrant). The index is based on the geometric mean of two proportions comparing the actual value to the benchmark value:

$$p = \sqrt{\frac{Actual\ Duration}{Benchmark\ Duration} * \frac{Actual\ Cost}{Benchmark\ Cost}}$$

We subsequently normalize this p to a value ranging from 0-100 with 100 being best via:

$$Cost\ Duration\ Index = \frac{(p_{\max -}\ p)}{(p_{max})} * 100$$

Stakeholder Satisfaction

Human satisfaction is a complex concept, involving many components such as physical, emotional, mental, social, and cultural factors (Wu, Naqibuddin, & Fleisher, 2001) (Pascoe, 1983). From behavioral science and consumerism multiple theories have emerged on psychometrically validated surveys on satisfaction (e.g. (Auquier, et al., 2005) (Atkinson, et al., 2004)). Although extended handbooks are available on the setup of a satisfaction survey (Hayes, 1998), we opted for a lean survey setup. The main reason to do so was a requirement from *BelTel* executives to make the survey as short as possible in order to minimize disturbance to the daily work for employees. An important argument for this requirement was the fact that the survey was implemented as a fixed part within the release process, meaning that some staff members had to fill it out several times during the release process (e.g. release managers that where involved in more projects that were included in one release filled out a separate survey for each individual project), or within every release (e.g. team members of Scrum-teams). We assume that this lightweight requirement will apply for other companies too and therefore is a precondition for a successful metric.

Stakeholder satisfaction is a measure of the satisfaction of stakeholders of a specific project with the way a project was performed and with the results as delivered by that project. *Stakeholder satisfaction* is measured by asking stakeholders of a specific project to rate their satisfaction on two aspects; the way a project was performed (the project's process), and with the results as delivered by a project (the project's result), for which we use questions with a 1 to 5 rating scale.

In both *BelTel* and *DutchCo* surveys were answered by internal stakeholders of projects: e.g. project managers, developers, testers, product owners. In case external stakeholders were included, these were working for *BelTel* or *DutchCo* as client or business analyst for a specific project. Due to the fact that *BelTel* did not want to involve external (real) customers in the study no external stakeholders in the meaning of end-users of a projects' deliverables were involved in the surveys.

Perceived Value

Value of software projects is a complex metric to measure (Shepperd, 2014), and studies are not specific on how they define value (Dingsøyr & Lassenius, 2016). It is difficult, if not impossible, to measure objectively and indisputable the real value as delivered by software projects to customers of *BelTel* and *DutchCo*. Is *real value* about money? Does it mean financial value, as in studies indicated by Return Of Investment (ROI) (Solingen, 2004)? Or is *real value* measured by *net promotor score* (NPS), as other studies indicate (Green, 2011) (Hofner et al., 2011) (Feyh & Petersen, 2013)? Such holistic measurements on value are often difficult to make for a single project, and they cannot easily be related to single software projects, mainly because too many different factors are of influence for such measurements.

To approximate the real value, we measure *perceived value* as a qualitative measure of the perception of stakeholders of each project. This is based on the notion that in fact every measurement is an agreement on a measurement procedure that sufficiently approaches the actual value (Solingen, 2004).

Perceived value is measured for each stakeholder in a specific project, on four aspects: *BelTel*'s or *DutchCo*'s customers, *BelTel* or *DutchCo*'s financials, *BelTel* or *DutchCo*'s internal process effectiveness, and *BelTel* or *DutchCo*'s innovation. We base the use of the four perspectives *customer*, *financial*, *internal process*, and *innovation* on the Balanced Scorecard (Kaplan & Norton, 1995). Based on the results per project of the four *perceived value* measures a *perceived value (overall)* is calculated, with the number of measures (not counting the choice "Don't know") as weighting factor.

7.3.5. Project Selection

Because we are particularly interested in data of finalized projects, all metrics are measured once a release is finalized, since only then we know the actual cost and duration of projects. Since we want to measure the effects of *stakeholder satisfaction* and *perceived value* on a software portfolio as a whole, we did not make any selection in the subset of projects within each release, except for the fact that we only selected projects that delivered software functionality (the projects could be counted in function points). Projects that do not include any software component (e.g. infrastructure projects or configuration projects) are excluded from our study.

7.3.6. Data Collection procedure

Collection of quantitative data

Within *BelTel*, a major part of the data collection for our case study was performed within the measurement capability that was already operational within the software department of the organization. Data collection on *project cost*, *project duration*, *number of defects*, *project size*, and calculation of both *estimation quality factor* metrics was performed by members of a measurement team that was supported (for performing function point counts (IFPUG, 2009)) by measurement staff of *IndSub-A*, *BelTel*'s main Indian supplier.

Different artifacts were used as a source for function point counting, depending from the availability per project (e.g. sets of functional documentation, user stories recorded in one of the Scrum backlog tools, architectural documents, project documentation, user manuals, or wireframes). All project data was stored in a measurement repository that was provided for our study. The lead author of the study was part of the *BelTel*'s measurement team.

In the *DutchCo* case, a dedicated research project was performed in order to collect and analyze data of software releases. The lead author of this paper performed the size calculations in retrospect for *DutchCo*. Due to this, it was possible to replicate the study that we performed within *BelTel* in exactly the same way in the *DutchCo* organization. All quantitative data was defined and collected in the same way. Function points were counted according to the same counting rules as used within *BelTel* (IFPUG, 2009). As a source for function point counting the user stories as recorded in the Scrum backlog tool were used.

Driven by the observations in our original study on correlations between *project cost* and *number of defects* on one hand, and *stakeholder satisfaction* on the other, we decided to collect data from finalized software releases within *DutchCo* in a more detailed way: cost data was categorized into a limited number of cost categories (e.g. design, build, test, deploy, management overall, quality assurance), and defect data was collected per defect severity (e.g. blocking, critical, high, medium, low).

Collection of qualitative (survey) data

Besides the project data that was collected as an operational practice, we collected data on *stakeholder satisfaction* and *perceived value*. To do so we conducted a questionnaire with stakeholders from *BelTel*, and later from *DutchCo*. The list of stakeholders was prepared in cooperation with the project managers of the applicable software projects, and consists of a mix of business and IT representatives that were involved in the subject projects. We asked the participants, who are stakeholders of a specific software project within a release, to rate their satisfaction with the way the project was performed and to rate their perception of the value that was added by the project. Besides ratings on a 1-5 rating scale we asked the participants to add free format text as an explanation of their perceptions. The questionnaire consists of five questions:

1. *What was your role in project Project_Name?*
2. *How satisfied are you with the way project Project_Name was performed (the project's process)?* (1-5 rating scale);
3. *How satisfied are you with the results of project Project_Name (the results as delivered by the project)?* (1-5 rating scale);
4. *How would you rate the delivered value of project Project_Name to the following aspects?* (1-5 rating scale, with 'Don't know' as an option; this choice was excluded from further analysis).
    a. *BelTel's Customers* (Value in terms of delivered to customers of *BelTel*);
    b. *BelTel's Financial* (Value in terms of financial revenue for *BelTel*);
    c. *BelTel's Internal Processes* (Value in terms of improvement and / or proper performance of *BelTel*'s internal processes);
    d. *BelTel's Innovation* (Value in terms of innovation of *BelTel*'s products or services delivered to its customers)?
5. *Are there any additional comments or suggestions you'd like us to know about this project?* (Free format text).

With regard to question 4: the additional information (between brackets) was shown to the participants when hovering with a mouse pointer over a question mark next to the text of each of the four aspects. Within *DutchCo* we applied the same electronic survey for stakeholders of the finalized software releases, including team members from the *IndSup-B* teams located in India.

7.3.7. Analysis Procedure

To explore potential relationships between the collected metrics, we tested for association between paired samples. Because all sample data is not normally distributed (see Table 7.3 for details on skewness and kurtosis and the boxplots in Figure 7.1), we used a Spearman rank correlation coefficient test for this purpose. In order to understand the underlying principles that can explain the outcomes of the quantitative analysis, we studied the free format text from the surveys.

Following (Hopkins, 2000), we prevent from Type I errors, e.g. finding a correlation by chance, simply because multiple comparisons are performed on the same dataset, by performing Bonferroni corrections on all p-values. We used an alpha of 0.05/26 (the number of projects in scope of this study), meaning that we assume all p-values above 0.0019 as not significant (Hopkins, 2000). Based on (Rumsey, 2016), we consider a significant correlation higher than 0.3 (or lower than −0.3) to be moderate, a significant correlation score higher than 0.5 (or lower than −0.5) to be strong, and a significant correlation above 0.9 (or lower than -0.9) to be very strong.

To compare the outcomes of the quantitative analysis of the project metrics with the survey we coded the free format text that resulted from the surveys that were performed within *BelTel* and *DutchCo*. We use the tool Qualyzer[4] for this purpose. We applied open coding, breaking down the survey data into first level concepts and second-level categories. Coding was performed by the author of this thesis, and reviewed by the other authors of the original research paper.

7.4. Results

7.4.1. Description of the *BelTel* Projects

Within the scope of our study we evaluated four software releases within *BelTel*, covering a total of 22 software projects. Table 7.2 gives a brief description of each project, where the numbering of the projects indicates in which

[4] http://qualyzer.bitbucket.org

Table 7.2. The BELTEL projects in scope of the case study.

Project ID	Project Description
BelTel 3.1	Rules- and regulations driven small Billing project
BelTel 3.2	Implementation of a control on a Billing application
BelTel 3.3	Release-based enhancements on CRM-application (Scrum)
BelTel 3.4	New campaign management tool (3rd part of a program)
BelTel 3.5	Release-based enhancements on a mobile App (Scrum)
BelTel 4.1	Enhancements on a Billing application
BelTel 4.2	Release-based enhancements on CRM-application (Scrum)
BelTel 4.3	Frontend project: Connect Google Play
BelTel 4.4	Rules & Regulations enhancement: fee for customers
BelTel 5.1	Release-based enhancements on CRM-application (Scrum)
BelTel 5.2	New campaign management tool (4th part of a program)
BelTel 5.3	Data warehouse 4 sprints of an iteration (Scrum)
BelTel 6.1	Enhancement to integrate payment by credit-card-aliases
BelTel 6.2	Enhancement to implement Apple Store code
BelTel 6.3	Release-based enhancements on CRM-application (Scrum)
BelTel 6.4	Adapt a procedure on an online platform
BelTel 6.5	E-invoice for a subset of customers in a Billing system
BelTel 6.6	Easy Script for cleanup of master MSISDN
BelTel 6.7	Rules & Regulations project on a Billing application
BelTel 6.8	Frontend enhancement: Shopper user interface e-services
BelTel 6.9	Once-only migration project
BelTel 6.10	New Order Management System (part of program, Scrum)

release each project was finalized and in which company a project or release was performed (e.g. *BelTel* 6.4 is a *BelTel* project that finalized in Release 6).

The software projects in scope represent a varied outline of *BelTel*'s software project portfolio. It includes projects of different business domains, sizes, cost patterns, durations, and delivery approaches. Some projects are typically once-only, with teams that were put together for the purpose of one project only. Others are part of subsequent iterations within a release structure with a steady heartbeat and a fixed, experienced team. Sixteen projects are characterized as plan-driven, while six followed a more agile (Scrum)

Table 7.3. Descriptive statistics of the BELTEL project data.

	Cost Duration Index	Project Cost (EUR)	Project Duration (Months)	Project Size (FPs)	Number of Defects
Minimum	86.92	8,000	4.96	12	1
First Quartile	93.27	44,001	8.37	25	3
Median	97.28	66,209	10.18	39	9
Third	98.57	118,876	11.73	126	23
Maximum	99.78	296,000	19.03	324	223
Mean	95.90	99,615	10.20	79	29
Skewness	-1.03	1.27	0.78	1.71	3.19
Kurtosis	0.06	0.77	1.43	2.71	10.89
St. Deviation	3.51	78,209	3.22	82	55

Project data (n = 22).

delivery approach, however a formal Scrum-by-the-book approach was not in place (i.e. sprints where performed, a backlog was managed and prioritized in a backlog tool, a product owner was in place, however no retrospectives were performed, no Scrum master was in place).

All projects were performed separately. Yet from the User Acceptance Testing onwards they were combined as a release deployed into *BelTel*'s production environment. Looking at the total cost of a release, on average 60% was spent on software projects. The remaining cost were spent on infrastructure projects, small innovations, and configuration projects, and as such do not fit into the *cost duration matrix* approach. These projects are out-of-scope for this case study.

Table 7.3 gives an overview of the descriptive statistics of the *BelTel* projects involved in the case study. As the table shows, the software projects in scope of the *BelTel* study are all relatively small in size, when compared to the projects in our research repository, ranging from 4 to 4600 Function Points (FPs): *project size* ranges from 12 FPs to 324 FPs, with a median of 39 FPs. To examine differences between the *BelTel* projects in scope of this study with our research repository as a whole, holding data of 492 software projects from

Table 7.4. Results from a Wilcoxon rank sum comparison of BELTEL releases (n = 22) with peer groups (n = 492).

	Median BelTel	Median peer group	W	p-value
Project Size	39	116	7148	0.0108
Project Cost	66,209	278,156	8003	0.0001
Project	10.18	8.41	4008	0.0394
Number of	9	72	2989	0.1028
Cost per FP	1612	2520	6952	0.0239
Days per FP	7.85	2.08	3058	0.0006
Defects per FP	0.22	0.16	2280	0.6621

The in light grey highlighted rows indicate statistically significant difference when applying Bonferroni corrections based on 22 comparisons, at the overall level of significance of 0.05 (we assumed all p-values above 0.0023 as not significant).

four different companies, we performed Wilcoxon ranked sum tests with Bonferroni corrections to compare overall differences, and differences per size (see Table 7.4).

If the data were sampled from a population with the median of the research repository, one would expect the sum of signed ranks (in the table reported as W) to be relatively small. The comparison shows that *BelTel* significantly differs from the other projects in the repository on *project cost*, as well on *days per FP*. On all other metrics no significance was found in the test. With regard to *project cost* we see this effect also in the boxplots in Figure 7.1; *BelTel* clearly shows overall lower cost for its projects compared to the other companies in our research repository. Although not confirmed by the statistical tests, a similar effect can be seen for *project size*; the boxplots indicate that overall *project size* for *BelTel* projects is smaller than that of the other companies. An explanation for differences in the outcomes of statistical tests and the boxplots in Figure 7.1 might be that the first only includes the 22 *BelTel* projects that are in scope of this study, while the second includes all 157 *BelTel* projects from our research repository.

Besides project metrics, we collected data of the *BelTel* projects on *stakeholder satisfaction* and *perceived value* by sending an online questionnaire to applicable stakeholders of each software project once the technical go live was performed. The overall completion rate of all surveys within *BelTel* was

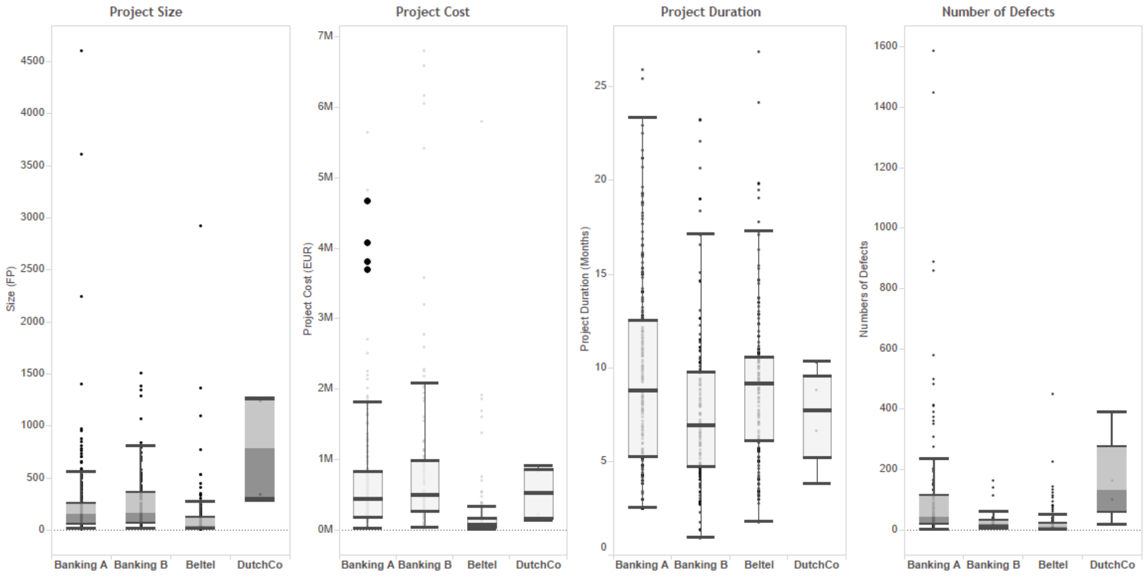


Figure 7.1. Boxplots of resp. project size, project cost, project duration, and number of defects of four organizations that are incorporated in our research repository of 492 projects. The boxplots indicate that DUTCHCO *projects significantly deviate on project size and number of defects from projects from other companies in our research repository, and not as such on project cost and project duration.*

69%. Over a period of four releases 103 surveys were completed by 53 individual respondents. One respondent could answer surveys for different projects in one release, or repeated surveys for a series of iterative projects over different releases.

7.4.2. Description of the *DutchCo* projects

Within the scope of our study we examined four *DutchCo* releases, all built from a large number of user stories. Table 7.5 gives a brief description of each release, where the numbering of the releases indicates in which company a release was performed; e.g. *DutchCo* 5.1 AM is a *DutchCo* release that was applicable to the asset management (AM) component of its billing solution.

Unlike *BelTel*, where the software portfolio includes a mix of projects of various business domains, delivery models and governance structures, the portfolio of *DutchCo* is more heterogeneous in nature. *DutchCo* implements only four releases each year to its customers. Due to that, these releases are

Table 7.5. The DUTCHCO projects in scope of the case study.

Project ID	Project Description
DutchCo 5.1 AM	Release containing asset management (AM) user stories.
DutchCo 5.1 POR	Release containing portal (POR) user stories.
DutchCo 5.2	Release applicable on DutchCo's general billing solution.
DutchCo 5.3	Release applicable on user stories for customer VF/BK.

All project data of DutchCo is to be found in a technical report (Huijgens et al., 2017e).

usually quite large in size. All *DutchCo* releases relate to the same business domain, namely the billing solution it provides to its customers. However, it occurs that different sets of functionality are delivered to customers, due to differences in requirements.

DutchCo's user stories are maintained in its backlog management tool, and continuously bundled in sprint backlogs. As a result, the governance structure of *DutchCo* is relatively simple. There are no projects, and there is a limited budget and planning activity. *DutchCo* has adopted a Scrum approach. Scrum teams are organized by functional component (e.g. Portal, Asset Management, Reporting, and Database). A large part of the Scrum teams is working from India, managed by *IndSup-B*.

Table 7.6 gives an overview of the descriptive statistics of the four *DutchCo* releases involved in this extended case study. The *DutchCo* subset exists of four releases, two relatively smaller ones (although still as large as *BelTel*'s largest projects), and two large ones. As can be seen in Figure 7.2 and in Table 7.6 above, the two oldest *DutchCo* projects (*DutchCo* 5.1 - POR and *DutchCo* 5.1 – AM), are smallest in size (resp. 277 and 335 FPs). However, both are comparable to the largest projects from the *BelTel* case. The two newest *DutchCo* releases are relatively large, compared to the projects in the *BelTel* subset: *DutchCo* 5.2 is 1233 FPs in size, and *DutchCo* 5.3 is 1261 FPs in size. Apparently driven by a schedule of four deployments per year, combined with a tendency to bundle the user stories of all functional components of its system, *DutchCo* releases tend to grow relatively large.

We performed a Wilcoxon rank sum comparison with Bonferroni corrections between the *DutchCo* subset and our research repository as a whole,

Table 7.6. Descriptive statistics of the DUTCHCO *project data.*

	Cost Duration Index	Project Cost (EUR)	Project Duration (Months)	Project Size (FPs)	Number of Defects
Minimum	95.03	125,827	3.78	277	15
First Quartile	96.12	192,040	5.90	321	78
Median	96.81	514,486	7.71	784	131
Third	97.41	835,343	9.18	1240	219
Maximum	98.22	896,788	10.29	1261	386
Mean	96.72	512,987	7.37	777	166
Skewness	-0.147	-0.002	-0.209	-0.003	0.434
Kurtosis	-1.952	-2.404	-2.054	-2.432	-1.865
St. Deviation	1.33	399,020	2.832	543	158

Project data (n = 4).

holding data of 492 software projects from four different companies, to compare overall differences, and differences per size (see Table 7.7 and Figure 7.1). The comparison shows that *DutchCo* significantly differs from the other projects in the repository on *project size*, as well on *cost per FP* and *days per FP*. On all other metrics no significance was found in the test.

We observe two findings here. Firstly, *DutchCo* releases have on average a larger size than other projects in our repository, which is good. This leads to a positive effect, from benchmarking purposes; due to the larger size of *DutchCo* releases also *cost per FP* and *days per FP* are better than the values of the other companies in our research repository.

Although no statistical evidence is found for any differences between the *number of defects* of both distributions (see Table 7.7), the boxplot view in Figure 7.1 indicates that besides *project size*, *DutchCo* also deviates from its peer groups on *number of defects*. Based on this we assume that a good score on *project size* might be counterbalanced here by a bad score on *number of defects*.

Besides project metrics as described above, we collected data on *stakeholder satisfaction* and *perceived value* by sending an online survey to applicable stakeholders of each software release once the technical go live was performed. For this purpose, we used the same electronic survey that was used

Table 7.7. Results from a Wilcoxon rank sum comparison of DUTCHCO releases (n = 4) with peer groups (n = 492).

	Median DutchCo	Median peer group	W	p-value
Project Size	784	116	219	0.0074
Project Cost	514,486	278,156	808	0.5387
Project Duration	7.71	8.41	1135	0.5981
Number of Defects	131	72	240	0.0564
Cost per FP	686	2520	1791	0.0047
Days per FP	0.28	2.08	1789	0.0048
Defects per FP	0.22	0.16	565	0.8762

The in light grey highlighted rows indicate statistically significant difference when applying Bonferroni corrections based on 4 comparisons, at the overall level of significance of 0.05 (we assumed all p-values above 0.0125 as not significant).

before within *BelTel*. The overall completion rate of all surveys within DutchCo was 71%. Thirty (30) surveys were completed by 30 individual respondents of both *DutchCo* and *IndSup-B*. Due to the fact that the three first releases were measured relatively long after finalization of each release, only for the latest *DutchCo* release an electronic survey was performed.

7.4.3. Results of plotting on the Cost Duration Matrix

We used the model that we developed in previous research to compare a portfolio of projects to the benchmark, by means of a *cost duration matrix*, as shown in Figure 7.2 for the 26 projects under study in this chapter. A detailed description of the *cost duration matrix* can be found in Section 2.3.2.

As can be seen from the figure, most of the 26 projects in the portfolio are cheaper than the benchmark would predict (right of the 0%-cost bar), yet take longer than expected (below the 0%-duration bar). The 0%-lines divide the *cost duration matrix* into four quadrants:

1. *Good practice* (top right): projects that score better than average for both cost and duration. In Figure 7.2, there are six projects in this quadrant, of which three of *BelTel* (5.3, 4.2, and 3.5) and three of *DutchCo* (5.1 - AM, 5.2, and 5.3).

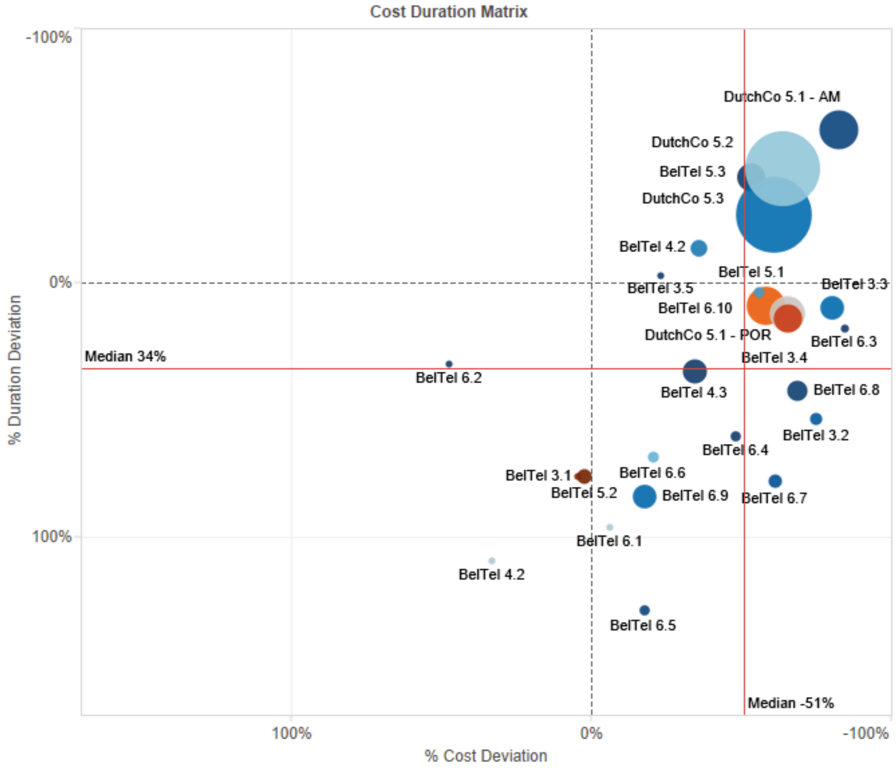


Figure 7.2. A cost duration matrix showing the 22 BELTEL and 4 DUTCHCO projects that are subject of the study.

2. *Cost over time* (bottom right): projects that score better than average for cost, yet worse than average for duration. This is where the majority of projects are in Figure 7.2.
3. *Bad practice* (bottom left): projects that score worse than average for both cost and duration. In Figure 7.2, there are four projects in this quadrant, all from *BelTel*.
4. *Time over cost* (top left): projects that score better than average for duration, yet worse than average for cost. In Figure 7.2, there are no projects in this quadrant.

The overall performance of the portfolio is furthermore summarized through the two red 'median' lines: On average, projects in the subject portfolio take 34% more time than expected from the benchmark, yet are 51%

cheaper. The *cost duration matrix* provides a tool to compare two project portfolios in terms of *project cost* and *project duration*. Our comparisons are based on the benchmark of 492 projects from the finance and telecom industries, described in more detail in (Huijgens et al., 2014c) (Huijgens et al., 2015c). The benchmark of 492 projects contain 157 previous projects from *BelTel*, and 4 previous ones from *DutchCo*, making it a suitable benchmark to compare the new additional 26 projects against.

7.4.4. Results of the tests for association

To identify potential relationships between the different metrics that we collected we performed a series of tests on paired samples of each metric, by using Spearman rank correlation coefficient. Because for only one *DutchCo* release data on *stakeholder satisfaction* and *perceived value* was measured (for only the latest release a survey was performed), we decided to test for associations on the *BelTel* and *DutchCo* dataset as a whole.

The results of these tests are shown in Table 7.8. The table is setup in the form of a matrix that pairs sets of two metrics. For each pair the correlation coefficient is shown, including (between brackets) the associated p-value. A color indicates significant correlation: dark grey indicates a strong (positive or negative) linear relationship, bright grey indicates a moderate linear relationship, light grey indicates a weak linear relationship. Results of the tests for association on the *BelTel* projects only can be found in Table 3 of the original research paper (Huijgens et al., 2016c).

We counteracted the problem of multiple comparisons by performing a Bonferroni correction. We compared each individual p-value to its Bonferroni critical value, (i/m)Q, where i is the rank, m is the total number of tests, and Q is the false discovery rate. We used 0.10 as false discovery rate, according to (McDonald, 2016). The largest p-value that has p<(i/m)Q is significant, and all of the p-values smaller than it are also significant, even the ones that aren't less than their Bonferroni critical value. A color indicates samples that are correlated: dark grey indicates a very strong (positive or negative) linear relationship (correlation coefficient higher than 0.70), moderate grey indicates a strong linear relationship (correlation coefficient between 0.50 and 0.70). Significant samples with a correlation coefficient lower than 0.50 are indicated in light grey.

However, a remark on the way we interpreted the results in Table 7.8 is in place. If results relating to the previous *BelTel* analysis agree with the results including the *DutchCo* data, we assume that both organizations are exhibiting similar results. If the results are completely different when the *DutchCo* results are included, we conclude that the companies are behaving differently, and that further research is needed to establish whether the new combined results are valid.

A second warning is in place with regard to some of the metrics we use. As it is dubious practice to correlate metrics that have a functional relationship between them (e.g. *cost per FP* and *days per FP*), as likely spurious correlations are found (Jørgensen, Halkjelsvik, & Kitchenham, 2012), we do not valid any findings with regard to these metrics as reliable. Analysis of the statistical tests for association between paired samples as depicted in Table 7.8 results in the following observations.

Observation 1: Strong positive correlations are found between project size, project cost, and number of defects.

In the first column of Table 7.8, it can be seen that *project size*, measured in function points, is strongly associated with *project cost* and *number of defects*. This effect is known from related studies (Huijgens et al., 2014c) (Boehm, 1984) and as such not a surprise in our research. The second column shows that also among themselves *project cost* and *number of defects* are strongly interrelated. However, where in many other organizations a clear correlation is found between *project size* and *project duration*, both *BelTel* and *DutchCo* show an atypical pattern. *Project size* and *project duration* are not related in any way.

This is the case when both *BelTel* and *DutchCo* are analyzed in a combined way, like inventoried in Table 7.8 and plotted in Figure 7.3, but also when examined separated they both show this effect (*DutchCo* shows a p-value of 0.9167 and a correlation coefficient of -0.2, the fact that only 4 releases are included in this analysis makes this test rather unreliable.

In order to examine whether this effect is only linked to the set of 26 projects in scope, or whether this effect goes for *BelTel* as a whole, we perform the test also with the *BelTel* projects that are not included in this chapter, yet available in our repository.

Table 7.8. Matrix with test results of association between paired samples, using Spearmans's rank correlation coefficient.

	Project Size	Project Cost	Project Duration	Number of Defects	Cost per FP	Days per FP	Defects per FP	Cost Duration Index
Project Cost	0.81 (0.000							
Project Duration	-0.13 (0.53)	0.09 (0.65)						
Number of Defects	0.68 (0.000	0.70 (0.000	-0.22 (0.32)					
Cost per FP	-0.70 (0.000	-0.19 (0.35)	0.35 (0.08)	-0.36 (0.10)				
Days per FP	-0.96 (0.01)	-0.72 (0.000	0.35 (0.08)	-0.66 (0.001)	0.73 (0.000			
Defects per FP	-0.14 (0.54)	0.10 (0.67)	0.05 (0.83)	0.59 (0.004	0.39 (0.07)	0.15 (0.50)		
Cost Duration Index	0.09 (0.66)	-0.03 (0.88)	-0.01 (0.95)	0.18 (0.43)	-0.24 (0.22)	-0.06 (0.76)	0.03 (0.89)	
Stakeholder Satisfaction (Process)	0.02 (0.93)	-0.26 (0.24)	-0.44 (0.04)	-0.30 (0.23)	-0.36 (0.10)	-0.21 (0.34)	-0.50 (0.04)	0.01 (0.97)
Stakeholder Satisfaction (Result)	-0.00 (0.99)	-0.16 (0.48)	-0.47 (0.03)	-0.07 (0.78)	-0.27 (0.22)	-0.21 (0.35)	-0.23 (0.35)	-0.04 (0.86)
Perceived Value (Overall)	0.32 (0.15)	0.07 (0.75)	-0.10 (0.68)	0.04 (0.89)	-0.34 (0.13)	-0.37 (0.10)	-0.20 (0.44)	0.04 (0.86)
Perceived Value (Customer)	0.34 (0.13)	0.10 (0.68)	-0.09 (0.69)	0.06 (0.82)	-0.35 (0.12)	-0.38 (0.09)	-0.20 (0.43)	0.04 (0.85)
Perceived Value (Process)	0.26 (0.26)	0.01 (0.98)	-0.03 (0.89)	-0.06 (0.82)	-0.27 (0.24)	-0.29 (0.20)	-0.15 (0.56)	0.06 (0.78)
Perceived Value (Financial)	0.32 (0.15)	0.07 (0.75)	-0.10 (0.68)	0.04 (0.89)	-0.34 (0.13)	-0.37 (0.10)	-0.20 (0.44)	0.04 (0.86)
Perceived Value (Innovation)	0.34 (0.13)	0.10 (0.68)	-0.09 (0.69)	0.06 (0.82)	-0.35 (0.12)	-0.38 (0.09)	-0.20 (0.43)	0.04 (0.85)
Estimation Quality Factor (Cost)	-0.08 (0.78)	0.02 (0.98)	-0.03 (0.92)	-0.32 (0.37)	0.11 (0.74)	-0.03 (0.93)	-0.28 (0.43)	0.13 (0.68)
Estimation Quality Factor (Duration)	-0.36 (0.10)	-0.43 (0.05)	-0.30 (0.01)	-0.18 (0.45)	0.16 (0.48)	-0.23 (0.31)	0.00 (0.99)	-0.25 (0.26)

The table above shows results from a test of association between paired samples of the 26 software projects from both case studies, using Spearman's rank correlation coefficient. Due to the fact that the DutchCo case contained a limited number of four projects, we performed the association tests over the total set of 26 projects from both BelTel and DutchCo. The overview shows for each test the correlation coefficient and between brackets the p-value.

Table 7.8. Continued.

	Stakeholder Satisfaction (Process)	Stakeholder Satisfaction (Result)	Perceived Value (Overall)	Perceived Value (Customer)	Perceived Value (Process)	Perceived Value (Financial)	Perceived Value (Innovation)	Estimation Quality Factor (Cost)
Project Cost								
Project Duration								
Number of Defects								
Cost per FP								
Days per FP								
Defects per FP								
Cost Duration Index								
Stakeholder Satisfaction (Process)								
Stakeholder Satisfaction (Result)	0.72 (0.0							
Perceived Value (Overall)	0.09 (0.71)	0.24 (0.30)						
Perceived Value (Customer)	0.07 (0.77)	0.23 (0.31)	0.99 (0.000					
Perceived Value (Process)	0.10 (0.66)	0.20 (0.37)	0.98 (0.000	0.96 (0.000				
Perceived Value (Financial)	0.09 (0.71)	0.24 (0.30)	1.00 (0.000	1.00 (0.000	0.98 (0.000			
Perceived Value (Innovation)	0.07 (0.77)	0.23 (0.31)	0.99 (0.000	1.00 (0.000	0.94 (0.000	0.99 (0.000		
Estimation Quality Factor (Cost)	-0.02 (0.96)	0.16 (0.62)	-0.23 (0.51)	-0.23 (0.51)	-0.23 (0.50)	-0.23 (0.51)	-0.23 (0.51)	
Estimation Quality Factor (Duration)	0.35 (0.12)	0.20 (0.38)	-0.32 (0.16)	-0.35 (0.13)	-0.28 (0.24)	-0.32 (0.16)	-0.36 (0.13)	-0.35 (0.26)

A color indicates samples that are correlated: dark grey indicates a very strong (positive or negative) linear relationship (correlation coefficient higher than 0.70), moderate grey indicates a strong linear relationship (correlation coefficient between 0.50 and 0.70). Significant samples with a correlation coefficient lower than 0.50 are indicated in light grey.

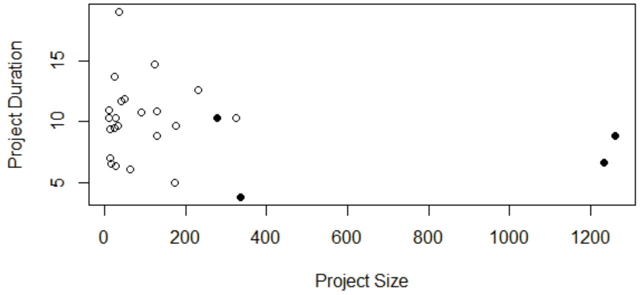


Figure 7.3. Plot of project duration versus project size; BelTel projects are indicated in open dots, DutchCo in closed dots.

A test with all 157 *BelTel* projects included shows a p-value of 0.002, and a correlation coefficient of 0.24, indicating that also in this case no correlation between *size* and *duration* is found. This outcome supports our observation that regardless the size of a project the duration is typically ten months. This is confirmed by a relatively low standard deviation for *BelTel*'s *project duration* (see Table 7.3).

In spite of this atypical effect with regard to *project duration*, in the fifth row a strong correlation can be seen between *days per FP* and *cost per FP*. Besides that we observe a relation between *days per FP* and *project size* and *project cost*. However, due to the functional relationship between both metrics we do not valid these findings as reliable (Jørgensen, Halkjelsvik, & Kitchenham, 2012).

Observation 2: Stakeholder satisfaction for both process and result are strongly interrelated to each other. Stakeholder satisfaction relates negatively with project duration.

Row nine of Table 7.8 shows that an observation that was found in our original study (Huijgens et al., 2016c) with regard to *stakeholder satisfaction*, remains intact. Both satisfaction ratings for process and product correlate strongly with each other. The fact that the same results are found for *BelTel* alone, and also when the *DutchCo* data is added provides evidence that the observation applies to both companies and may represent a more general observation that high satisfaction ratings on process link with high ratings on the delivered product. However, the weak correlation between *project duration* and *costs per FP* and *days per FP* was not visible in the *BelTel* data and has only occurred with the addition of the *DutchCo* data. This suggest the effect is due to the *DutchCo* data (based on a single large release) and calling for more research to investigate whether the effect is real.

Column three shows that *project duration* has a moderate negative relation with *stakeholder satisfaction* for both *process* and *result*. Longer project durations tend to lead to lower satisfaction rates. Furthermore *project duration* relates weakly with *cost per FP* and *days per FP*, indicating that longer project durations lead to higher *cost per FP* and a higher number of *days per FP*. However, due to the functional relationship between both metrics we do not valid these findings as reliable (Jørgensen, Halkjelsvik, & Kitchenham, 2012).

Two observations are related to *perceived value*. A weak positive linear relationship between *project size* and *perceived value (overall)*, as shown in our original study, is found here too, indicating that *perceived value* is higher for larger projects (in function points). Furthermore, several of the perceived value metrics shows weak negative relations with *cost per FP* and *days per FP*, indicating that lower cost and duration per FP links with higher scores on *perceived value*. This effect is much reduced compared with the original study with *BelTel* data only (Huijgens et al., 2016c), suggesting that it is a *BelTel* phenomenon.

A major limitation here is, that the *DutchCo* project for which *perceived value* and *stakeholder satisfaction* is measured, is significantly larger in size than all other *BelTel* projects. Figure 7.4, with on the X-axis the *project size* in function points, and on the Y-axis the *overall perceived value* rating of each project, clearly shows that a good comparison in fact is not yet possible in this context; more data is needed, especially from relatively larger projects. An additional remark on this phenomenon is that it may be that the correlations would have been weaker for *BelTel* in the original paper if that analysis had been based on a more robust correlation coefficient.

We observe a striking correlation between all mutually *perceived value* measurements. We assume that the four aspects are measuring the same construct, or that the answers to those items were influenced in the same way. This effect was not measured this strongly with *BelTel* data only. We assume

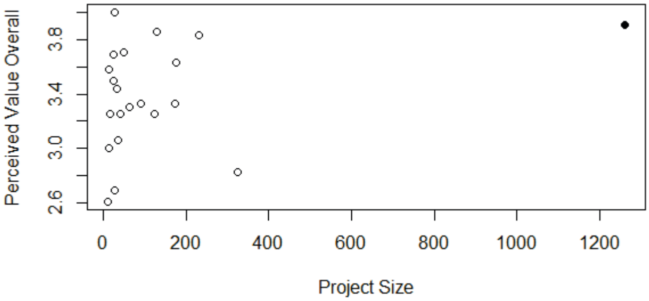


Figure 7.4. Plot of perceived value overall versus project size; BELTEL projects are indicated in open dots, DUTCHCO in closed dots.

the effect found now is an artefact of adding *DutchCo* data. Due to the fact that the results are unstable, we do not value these outcomes to high though. Other observations with regard to perceived value, as mentioned in our original paper (Huijgens et al., 2016c), seem to have vanished in this study. After adding the *DutchCo* data to the comparison, no relations with another project metric are observed.

A comparison of the results of the test for association which listed only *BelTel* results - see Table 3 in the original study (Huijgens et al., 2016c), with the results of the test in which both *BelTel* as *DutchCo* projects are included (see Table 7.8) - shows that the latter shows a clearer and more coherent pattern. Where the original, *BelTel* only, table shows a rather scattered pattern, the actual results focus on the three observations mentioned above.

Especially the statistical power of function points as a measure for *project size* stands out. Besides that, we found indications for a positive relationship between both *stakeholder satisfaction* measures, and between *stakeholder satisfaction* for *results* and *project duration*. We did not find direct evidence for strong relations between *perceived value*. However, we do have expectations with regard to this for future research due to a very strong interconnection between the four *perceived value* measures. In the next Subsection we challenge our observations by linking them to the free format text that resulted from the surveys that are performed at closure of each release.

Observation 3: Weak correlations are found between estimation quality factor for duration on the one hand and project duration and stakeholder satisfaction on the other.

A final observation that results from the quantitative analysis is about the quality of estimations with regard to project duration (see the bottom horizontal row in Table 7.8). When compared to the initial *BelTel* study, the only consistent observations are the negative correlation between *EQF (duration)* and *project duration* and the positive correlation between *stakeholder satisfaction* and *EQF (duration)*. The first correlation suggests that shorter projects are less well estimated with regard to duration. However, this effect was not visible in the analysis of the *BelTel* data, so must be due to the *DutchCo* data. The second suggests that stakeholders like accurate duration estimates, although in the initial study stakeholders were satisfied about the result, while after adding data stakeholders were satisfied about the process.

7.4.5. Results of the free format text analysis

In order to compare the outcomes of the quantitative analysis of the project metrics with the survey we coded the free format text that resulted from the surveys that were performed within *BelTel* and *DutchCo*. See Table 7.9 and Table 7.10 for the outcomes of the coding of *BelTel* and *DutchCo* free format text data. Both tables are ordered on the number of times a code was applied in the comments. We discovered seven main themes: In the following Subsections we discuss these main themes, where we combined connected coding aspects into one theme. A subset of comments given by participants from the surveys is included in the following Subsections, indicated by the letter “B” (for *BelTel*) or “D” (for *DutchCo*) followed by a participant number.

Quality, Deployment and Testing (A1, A3, A7)

The first thing that strikes us when looking at the results of the coding process is that aspects with regard to quality are high on the list of items that apply to the stakeholders. Most remarks were about good quality, however, a number had to do with low quality issues of deliverables.

A large number of negative comments given in the survey was related to the deployment of projects within a release into *BelTel*’s or *DutchCo*’s production environment. Most had to do with issues that occurred during this process (e.g. problems with environments or incidents in production that

Table 7.9. Results of the analysis of free format text of the BelTel survey.

	Points of attention for satisfaction and value	Count
A1	Quality (good quality 27, bad quality 12)	39
A2	Communication (good communication 21, bad communication 17)	38
A3	Deployment (issues with implementation 19, issues in production 9, bad or delayed implementation 9)	37
A4	Requirements (requirements not clear 15, good requirements 9, requirements creep 5, bad documentation or design problems 4)	33
A5	Stakeholders (satisfied stakeholders 29, low stakeholder involvement 3, unsatisfied stakeholders 1)	33
A6	Duration (good estimation of duration; in-time delivery 16, bad estimation of duration 7)	23
A7	Testing (good testing or good test environment 8, problems with testing 9, delayed testing 3	20
A8	Process (smooth, lean, or mature process 11, (agile) process needs improvement 3, bad process 2, process not according to standards 2)	18
A9	Project Management (scope problems 10, good project management 3, scope delivered 3)	16
A10	Agile Development (good product owner 4, good backlog management tool 2, use of tools unclear 1, agile process needs improvement 1, traditional release in agile process 1)	14
A11	Supplier Management (issues with supplier 11, good relation with supplier 1, bad alignment between parties 1)	13
A12	Team Aspects (good team spirit 7, team not fixed 1)	8
A13	Release Management (bad alignment project and release 6, release delayed 1)	7
A14	Value Aspects (good value 2, issues with value 4)	6
A15	Cost Aspects (within time and budget)	1

Table is sorted on Count.

needed to be fixed, repeated rollback of releases, improvements to be made in the deployment process, and in solving issues.

An explanation for the fact that many issues occur after going technically live is that *BelTel* uses the first week (or sometimes a longer period) to test deployments in the production environment. Usually projects are not commercially live during that period. Comments with regard to testing are often related to these deployment issues. Also here we find a majority of comments

Table 7.10. Results of the analysis of free format text of the DutchCo survey.

	Points of attention for satisfaction and value	Count
A7	Testing (issues and defects late in the release 10, testing to be improved 8, improve acceptance criteria 3, large number of defects 2)	23
A14	Value Aspects (innovative solution, yet customer specific 15, many new features, good business value for the customer 7)	22
A4	Requirements (changes in scope and unclear features 9, hidden business rules as cause for problems 9, improve application 3)	21
A5	Stakeholders (satisfied stakeholders 3, customer driven 17)	20
A2	Communication (good communication between teams 5, to be improved communication between teams 3, better communication with customer 5)	16
A6	Duration (time pressure at the end of a release 11, strict timelines 2)	13
A8	Process (the internal process needs improvement, but people notice the company is changing for the good too 10, no proper process in place 2)	12
A1	Quality (satisfaction about the delivered results 8)	8
A12	Team Aspects (cooperation between teams, team performance, high pressure on team 7)	7
A9	Project Management (poor communication about scope of a release 5)	5
A15	Cost Aspects (profit made 1, paid by customer, yet generic 1)	2
A10	Agile Development	1
A3	Deployment	0
A11	Supplier Management	0
A13	Release Management	0

Table is sorted on Count.

that are related to issues with test environments and the test process itself, for example:

'A lot of discussion on how we need to test...' (B39).

'There were some defects in production' (D30).

We note that deployment itself is not mentioned by any of *DutchCo*'s stakeholders. Maybe an explanation for this is the fact that (unlike the *BelTel* approach with centralized deployment by a separate release team) *DutchCo* teams are themselves responsible for deployment of solutions. Summarizing, for both *BelTel* and *DutchCo* a generic observation applies on quality:

Observation 4: Satisfied stakeholders tend to emphasize good quality, while dissatisfied stakeholders say testing and deployment need improvements.

Communication (A2)

The second most mentioned point on the stakeholder's list in both studied companies is about communication. A number of remarks have to do with good communication between parties. A remarkable finding within *BelTel* was that positive remarks all were related to external suppliers in the frontend development of website and app development, and not with the main strategic supplier *IndSup-A*. In the *DutchCo* case many positive remarks on commination had to do with team aspects, such as:

'Improvements in the cooperation between teams. A lot of work done in a short amount of time' (D02).

However, not all is well with communication. Besides the many positive remarks, there are also suggestions for improvement, sometimes related to the agile process:

'Communication and involvement for agile items is limited to the bare minimum, so the added value of release management is not really big here. The whole agile process is still pretty blurry to most of its stakeholders, so this definitely needs to be improved' (B48).

In the *DutchCo* case many of the negative responds on communication where about bad communication between teams. Overall, a generic observation can be made on communication:

Observation 5: Satisfied stakeholders emphasize good communication. Dissatisfied stakeholders say communication needs to be improved.

Requirements (A4)

Most of the comments related to requirements were about unclear requirements that hinder a project's progress, such as:

'Interpretation from requirements can be different and cause issues at testing phase' (B40).

A limited number of comments were made on bad documentation, design problems and requirements creep, but also some comments were made on the availability of good requirements.

In the specific *DutchCo* case several remarks were made on unclear or hidden business rules as a cause for problems. Yet, a generic observation can be made with regard to requirements in both studied companies:

Observation 6: Dissatisfied stakeholders emphasize unclear requirements, bad documentation, hidden business rules, and requirements creep.

Stakeholder Satisfaction and Duration (A5, A6)

Many of the comments related to stakeholder aspects were about satisfied stakeholders. Most comments had to do with the quality of delivery and the time-to-market of delivery. Project duration and time-to-market was mentioned by many participants, where most comments are about on-time delivery. In the *BelTel* case the following observation applied:

Observation 7: Satisfied stakeholders comment about good quality of duration estimates. Dissatisfied stakeholders comment about long duration and schedule overrun.

In the specific *DutchCo* case many respondents indicate they are satisfied with the product that was delivered to *DutchCo*'s customer, as for example stated by D01:

'Despite the time constraints a decent product was delivered'.

With regard to this some respondents mentioned the inclusion of the customer as a positive factor, as stated by D10:

'I am satisfied about the fact that we included the customer and got their feedback, which resulted in a better product'.

However, with regard to *Project Duration* a number of *DutchCo* respondents mention a high time pressure, especially towards the end of the release, leading to issues and defects in the last stages. As D13 says it:

'Time issues caused several problems'.

Although duration was mentioned by stakeholders from *DutchCo*, they did this from the perspective of time pressure at the end of the release. In the *DutchCo* case no estimations were prepared with regard to duration. However, in the original *BelTel* study we found that satisfied stakeholders comment about good quality of duration estimates, where dissatisfied stakeholders comment about long duration and schedule overrun. In the *DutchCo* study we found one additional observation with regard to time pressure:

Observation 8: Time pressure towards the end of a release leads to more defects.

Agile, Value, and Process (A10, A14, A8)

A more agile delivery process is one of the key innovations that are implemented within the software delivery organization of *BelTel*, as well in the *DutchCo* case. Knowing this we argue that the low number of comments related to this aspect by *BelTel* stakeholders (14) does not reflect the strategic choice of *BelTel* for a new delivery approach, including the investments made in coaching and implementing tools that support an agile way or working. Eight (8) comments were positive about the quality of the product owner and the backlog management tool in use.

However, some comments were related to the agile process itself that needed improvement, as stated for example by B48:

'The whole agile process is still pretty blurry to most of its stakeholders so this definitely needs to be improved'.

For an organization that made delivery of value a strategic innovation remarkably few comments were made on value aspects. Two were about good value being delivered, while most had to do with the lack of value, such as:

'No real feeling on the benefit of this project' (B45).

With regard to process aspects a limited number of comments were about needs for improvement, such as speeding up things and working in a more structured way. However, about as many comments were related to a lean and flexible process.

In the specific *DutchCo* case we observed that from the viewpoint of innovation stakeholders overall seem to be quite happy with the delivered result, although some respondents mention the fact that the release was only applicable to one specific customer, as for example stated by D21:

'This was a big step in innovation, which would help us move further old applications to new technology stack' (D21).

Some stakeholders mention that not only the delivered product is innovative, but also the applied internal process:

'New technology used, new groundwork for new applications has been set. New way of working also (introduction of design street, and QA-department)' (D10).

Observation 9: Satisfied stakeholders emphasize the delivery of good value to the customer.

However strongly related to remarks on to be improved internal communication, stakeholders feel that *DutchCo*'s internal process should be improved, although people see things changing for the good too, as for example mentioned by D01:

'The new structure of the company sort of has evolved on a better level'.

Although value certainly was addressed by *DutchCo*'s stakeholders, no comments were made related to the agile delivery approach. Where in the specific *BelTel* case an observation applied that the low number of comments related to agile processes does not reflect the company's strategic choice for a new delivery approach, we argue that the *DutchCo* omission is only partially comparable with the *BelTel* study; *DutchCo* stakeholders simply do not talk about agile because it is their only delivery approach. Thus, we adjust the *BelTel* observation to the following generic one:

Observation 10: 'Agile' itself is not always a point of discussion in companies, even when they are agile.

Supplier Management (A11)

A number of comments were about issues with suppliers, where also *BelTel*'s main supplier *IndSup-A* was mentioned several times, such as by B14:

'Very long delays and complete lack of knowledge and initiatives from Indsup'.

In the *DutchCo* case no specific comments were found that related to supplier management. In a way this is not surprising, since besides the *Ind-Sub-B* teams no external parties are applicable within *DutchCo*.

Cost

A remarkable observation is that only once a comment is made related to cost of projects in the *BelTel* study:

'Implementation as per time, budget, and quality' (B38).

No comments were made about the estimation accuracy with regard to project cost. The aspect of cost was mentioned only twice by *DutchCo* stakeholders.

'We got paid quite a sum of money for this and I think we made some profit' (D10).

'Solution was completely paid by customer where we can use this functionality as generic functionality in our own solution' (D03).

Although the backgrounds may be different, like *BelTel* cost seems not a big issue for *DutchCo* stakeholders:

Observation 11. Cost is mentioned by only one project stakeholder.

A warning is in place regarding interpreting frequency as importance. But frequency can also just mean salience; it's what was on participants' minds when we ask about it. So here, it may not be that cost is unimportant, it may be that few people thought to mention it.

7.5. Discussion

In order to validate the outcome of the quantitative analysis with the outcome of the qualitative analysis, we compared the observations from both analyses, as depicted in Table 7.11. We grouped the data into four themes that correspond to the horizontal rows in Table 7.8 that logically belong together: the core project metrics (*project size*, *project cost*, *project duration*, and *number of defects*), *stakeholder satisfaction*, *perceived value*, and the *quality of duration estimations*.

7.5.1. The Core Project Metrics

The strong positive correlations that we found between *project size* on the one hand and *project cost, project duration,* and *number of defects* (observation 1), confirm what is already known from related work (Huijgens et al. 2014c) (Boehm, 1984) (El Emam & Günes Koru, 2008) (Boehm et al., 2000a) (Heemstra & Kusters, 1991) (Bhardwaj & Rana, 2016). From this point of view, *project size*, measured in *function points*, can be considered as a very strong predictor of both cost and process quality.

Also the effect of project size as a risk factor is described earlier. Smaller projects tend to have lower cancellation rates (Rubinstein, 2007) (Sauer & Cuthbertson, 2003). Smaller projects tend to perform better in terms of quality, being on budget, and being on schedule (Rubinstein, 2007) (Sauer & Cuthbertson, 2003) (Sonnekus et al., 2004).

Project size is found to be an important risk factor for success (Barki, Rivard, & Talbot, 1993) (Jiang & Klein, 2000) (Schmidt et al., 2001) ((Zowghi

& Nurmuliani, 2002) (Heemstra & Kusters, 1989) (Chidambara et al., 2016). Note, however, that the literature does not match results from our study with regard to an economy-of-scale effect that larger projects in size are good, for *cost per FP* and *days per FP* (see Subsection 7.4.3). We argue that for most projects a trade-off is applicable between cost and duration on the one side, and risk on the other.

Despite the strong correlation, the use in practice of function points to measure project size suffers from shortcomings, such as additional training needed, subjective determination of complexity, and not considering the development environment (Singh et al., 2016).

In order to emphasize the effect of functional size (function points) as a normalizer we give an example related to the cheapest of all *BELTEL* projects versus the most expensive one. Project *BelTel* 6.3, a small (16 FPs) release-based enhancement on a CRM-application that was per-formed in a Scrum way as depicted above represents the minimum cost of 8,000 euro. This project scores good in the *cost duration matrix* in Figure 7.2, and shows the highest score of all for *stakeholder satisfaction* for both *process* and *result*.

To put things in perspective, the maximum cost of 296,000 euro is linked to Project *BelTel* 6.10, an implementation of a part of a new order management system. Yet, also this project scores on the upper side in the *cost over time quadrant*, mainly due to a high number of function points that are delivered; 324 FPs. This project also scores well for both *stakeholder satisfaction* and *perceived value*. We note that both projects were performed in a Scrum way as depicted above.

With regard to the strong correlations that we found between *project size* and other core project metrics such as *project cost*, *project duration*, and *number of defects*, it might be important to consider that *project size* was measured in function points, in a manual counting process. As described in the data collection approach, different artifacts were used as a source for function point counting, depending on the availability per project (e.g. sets of functional documentation, user stories recorded in one of the Scrum backlog tools, architectural documents, project documentation, user manuals, or wireframes). Manual counting was performed by different members of measurement teams of both companies, and reviewed by another member, to ensure proper use and interpretation of counting guidelines (IFPUG, 2009). In agile environments, where usually no upfront artifacts such as functional

Table 7.11. Summary of observations and implications for practice and research.

Themes	Observations from the quantitative analysis
The Core Project Metrics	Strong positive correlations are found between Project Size, Project Cost, and Number of Defects (observation 1).
Stakeholder Satisfaction	Stakeholder Satisfaction for both process and result are strongly interrelated to each other. (observation 2). Stakeholder Satisfaction relates negatively with Project Duration (observation 2).
Perceived Value	Although the different Perceived Value measures interrelate strongly with each other, we cannot draw general conclusions from this (observation 3). Weak correlations are found between Perceived Value and Project Size, Cost per FP, and Days per FP, we cannot draw general conclusions from this BELTEL phenomenon (observation 3).
Estimation Quality for Project Duration	Weak correlations are found between Estimation Quality Factor for Duration and Project Duration, and Stakeholder Satisfaction (observation 4).

and technical design documents are prepared, counting functional size can be challenging. However, we experienced in practice in both companies that descriptions of user stories in backlog tools together with additional information such as wireframes, where suitable to perform a reliable function

Table 7.11. Continued, right side of the table.

Observations from the qualitative analysis	Implications for practice and research
Cost does not seem an important issue for project stakeholders (observation 12). Time pressure towards the end of a release leads to more defects (observation 9) (DUTCHCO).	The strong correlations between Project Size on the one hand, and Project Cost, Project Duration, and Number of Defects on the other are confirmed by related work (Boehm, Software Engineering Economics, 1984) (Huijgens, van Solingen, & van Deursen, 2014) (El Emam & Günes Koru, 2008) (Boehm, Abts, & Chulani, 2000) (Heemstra & Kusters, 1991) (Bhardwaj & Rana, 2016).
Satisfied stakeholders emphasize good quality (observation 5), good communication (observation 6), and good quality of duration estimations (observation 8). Dissatisfied stakeholders state that testing and deployment needs improvement (observation 5), they emphasize unclear requirements, bad documentation, hidden business rules, and requirements creep (observation 7), they say communication needs to be improved (observation 6), and they comment about long duration and schedule overrun (observation 8).	Additional research is needed to identify how good quality of deliverables, good communication, and reliable estimations for duration can be controlled in practice in order to create satisfied stakeholders. Additional research is needed to identify how issues with testing and deployment, unclear requirements, hidden business rules, bad communication, and schedule overrun can be controlled in order to mitigate stakeholder dissatisfaction.
Satisfied stakeholders emphasize the delivery of good value to the customer (observation 10) (DUTCHCO). 'Agile' itself is not always a point of discussion in companies, even when they are agile (observation 11).	Additional research is needed (especially on medium and big sized projects) to validate whether larger projects tend to lead to higher perceived value scores. If this effect is true, functional size might be a potential indicator for value, that can be used in practice by Product Owners to prioritize (enterprise) backlogs.
Satisfied stakeholders emphasize good quality of duration estimations (observation 8). Dissatisfied stakeholders comment about long duration and schedule overrun (observation 8). Time pressure towards the end of a release leads to more defects (observation 9) (DUTCHCO).	Much research is performed on Effort Estimation of software projects, yet very limited research is performed on the effects of the quality of Duration Estimation, because the outcomes of this study indicate correlations with both stakeholder satisfaction and perceived value.

point count. In all cases so-called estimated Function Point counts were performed.

To automate the data collection process where possible, we strongly felt a need for some form of automated function point count, if possible based on the code itself. However, a follow-up exploratory study of 336 functional size measurement specialists that was performed based on this hypothesis did showed that overall automated functional size measurement was considered as important, but also difficult to realize (Huijgens et al., 2015b).

7.5.2. Stakeholder Satisfaction

Observation 2, indicates a moderate correlation between *stakeholder satisfaction* for both *process* and *result* and *project duration*. We found a moderate relation between *stakeholder satisfaction* for *estimation quality factor (duration)* too. This indicates that *stakeholder satisfaction* is related to interaction and being informed, yet also with conformance to planning and estimation of the delivery date. A strategy of 'no last minute surprises' as such helps better to increase *stakeholder satisfaction*, as well as giving attention to improvement of estimation and planning practices would.

Satisfaction of stakeholders with the development process and with the development outcome was studied before by Ferreira and Cohen (2008). They found "strong positive effects of agile practice (iterative development, continuous integration, collective ownership, test-driven design, and feedback) on stakeholder satisfaction with both development process and the project outcome". A relation between stakeholder satisfaction and with agile software development in an Indian context was found by Nazir et al. (2016).

Many participants mentioned communication to be important, while good communication is mentioned by satisfied stakeholders, and bad communication by dissatisfied ones. Approximately half of the comments were about good communication, such as good alignment between parties, good collaboration, and short feedback loops. The other half mention communication to be improved, such as provide information on processes and innovations (e.g. agile delivery), ongoing discussions, and miscommunication with suppliers. Unclear requirements, bad documentation, requirements creep, and bad quality of test and deployment resources are perceived by dissatisfied stakeholders as causes for bad quality of deliverables.

However, a warning is in place here: we notice that many positive comments on communication within *BelTel* also are linked to two specific Product Owners. We did not focus our research on roles within the subject projects, but this suggests that the fulfillment of a role by a specific person may be of greater influence on *stakeholder satisfaction* and *perceived value* than the subject delivery model. Note that this resonates with the first line of the Agile Manifesto: “Individuals and Interactions over Processes and Tools” (Beck, et al., 2001).

Contrary to the findings in our original study, we did not find evidence for correlations of *stakeholder satisfaction* with *number of defects*. However, in the qualitative analysis we do find many comments that are in one way or another related to those aspects. Quality of the deliverables (both good quality and to be improved quality), in combination with testing aspects and deployment into the production environment, is commonly mentioned in comments by all participants.

7.5.3. Perceived Value

We assume that the relative absence of comments that are related to the ongoing innovation of implementing a more agile delivery process within *BelTel*, in combination with the limited focus on value might be of importance here (observation 10). The low interest in agile innovation among *BelTel*’s stakeholders in a way reflects our findings in the quantitative analysis too. No significant relation is found between *perceived value* and any other project metric of software project deliverables. However, the limited number of projects in scope of this study, combined with the diversity in *project sizes* (with many small and only one large project) can be a reason for replicating our study with more data in future.

Furthermore, what strikes with regard to the value measures are the almost perfect correlation coefficients (from 0.94 to 0.99) with p-value < 0.001 that we found between all four *perceived value* measures mutually. This is not a good sign when designing scales. It might imply that the four aspects, derived from well-known research on Kaplan and Norton’s Balanced Scorecard (Kaplan & Norton, 1995), are measuring the same construct, or that the answers to those items were influenced in the same way. Besides that a warning is in place here with regard to functional relations between the perceived Value metrics. As it is dubious practice to correlate metrics that have a functional relationship between them, as likely spurious correlations are found (Jørgensen, Halkjelsvik, & Kitchenham, 2012), we do not consider the findings on interrelated *perceived value* metrics as reliable.

7.5.4. Estimation Quality for Duration

The comments given in the surveys confirm the more or less company specific observations with regard to *project duration*. Stakeholders of projects are satisfied when delivery of results is in-time, where we assume this relates to good quality of duration estimates. However, it needs to be said that the words *estimate* or *estimation* are never used in the comments. Dissatisfaction of stakeholders is often linked with too late delivery and long project durations (long waiting time).

In the *DutchCo* case we do not find direct evidence for this, although time pressure towards the end of a release, causing issues and defects at a late stage in the release, are mentioned as a source for problems by many respondents.

7.5.5. Success or failure: complex relations

Looking at the seventh row and eight column in Table 7.8, the test results of association between paired samples, it strikes that no correlation is found at all between *cost duration index* and any of the other samples. Apparently no relation exists between the position of a project in our *cost duration matrix* and the measure of *stakeholder satisfaction* and *perceived value* of that project.

Based on this we conclude that success and failure apparently are more complex than *cost*, *duration* and *defects* only: stakeholders can be satisfied or have the perception of much value delivered, even when a project is in the so-called *bad practice quadrant*.

We suspect, based partly on recent ongoing research, that the limitation in the current study to internal stakeholders of projects can play a limiting role here. For external stakeholders, usually the customers of the software organizations that actually pay for the software and use it in practice, cost, duration and number of errors seem to be an important factor for success or failure. Besides that we suspect that *perceived value* needs to be measured on a lower level than a project (e.g. at *user story* or at *epic* level). We argue that additional research is needed to unfold these complex relations.

7.5.6. Agile and Cost were not mentioned

Except for the last four themes that resulted from the quantitative and qualitative analyses, we found two issues that were not mentioned in the text free format of the surveys, and that were not found in the tests for association between paired samples.

Firstly, we found a low number of comments in both studies, that are related to the concept of *agile* itself. A number of observations that were applicable for *BelTel* did not apply to *DutchCo*, and the other way around. No evidence was found within *DutchCo* on the relation between satisfied stakeholders and good quality of estimates. *DutchCo* does not produce estimates as such for its software delivery activities.

Like *BelTel*, within *DutchCo* no specific remarks were made about the agility of its process. However, an agile development approach within *DutchCo* is widely assumed as the only form of process, no other approaches (e.g. plan-driven) are applicable. Thus, we assume that no comments were made about this just because it is 'business as usual'. However, as stated before frequency, needs not to be interpreted as importance.

Besides the development method used, we assume that also differences in test and deployment (e.g. release) approaches of both software companies influence the portfolio performances. The fact that *DutchCo* collects all of its features in large three-monthly releases, while *BelTel* runs a variety of software projects sequentially with eight combined releases per year, should be taken into account when comparing the performances of both companies.

A second finding was the fact that cost seems not an important issue for stakeholders within both *BelTel* and *DutchCo*. Only one comment by a *BelTel* stakeholder is made related to this. This finding applies to *DutchCo* too; not much attention is given in the comments on cost either. We think that this might be caused by the effect that in more or less agile organizations, the focus tends to shift from time and cost driven controlling towards scope and value driven steering. Agile teams tend to stay in place for longer periods, and budgeting often needs to be done only once a year, instead of many times per year in a pre-project phase in plan-driven organizations.

In a way this is not a large surprise in the *DutchCo* organization, since development teams are fixed and stay together for long periods of time. Due to this cost is just a derivative of effort spent by these teams. No budget estimations upfront are applicable in the *DutchCo* organization. Although this cannot be understood in a way that *DutchCo* is not interested in cost at all; *DutchCo*'s management team is highly interested in cost reduction based on the effects of shortening learning curves (Šmite & van Solingen, 2016) and efficiency improvements based on outsourcing approaches.

7.5.7. Implications

The outcomes of both our case studies might not simply be generalized to other environments. We identify a number of take-away-messages that apply to research and practice in other software companies too.

The first one relates to the from related work already known strong correlations between *project size* on the one hand, and *project cost*, *project duration*, and *number of defects* on the other, indicating the power of *project size* (measured in function points) as an indicator for cost, duration, and quality.

However, a link with agile development is poorly covered in research (Huijgens et al., 2015b), and in practice many agile software companies tend to see (manual) counting of functional size as waste. We argue that also agile practitioners and researchers should rethink and embrace *project size*.

Secondly, we argue on the one hand that good quality of deliverables, good communication, and reliable estimations for duration can be used to increase stakeholder satisfaction, and on the other that issues with testing and deployment, unclear requirements, hidden business rules, bad communication, and schedule overrun increase stakeholder dissatisfaction, and should therefore be mitigated when possible. Assuming that agile development methods might play a role here, additional research is needed to identify the backgrounds and ways to control these findings in a practical context.

Thirdly, we recognize a need for additional research (especially on medium and large sized projects) to validate whether larger projects tend to lead to higher *perceived value* scores. We think that measuring *perceived value* on a lower level than a software project, e.g. on user stories or epics, might result in other outcomes. If a strong positive effect is found in future, functional size might be a potential indicator for value, that can be used in practice by Product Owners to prioritize (enterprise) backlogs.

Finally, a fourth take-away-message relates to a need for additional research in a practical context on the effects of the quality of duration estimation, because the outcomes of this study indicate correlations with both *stakeholder satisfaction* and *perceived value*. Good quality estimation of a project's delivery date seems very important for stakeholders, and relates to the perception of value that is delivered.

7.6. Threats to Validity

7.6.1. Construct Validity

With regard to construct validity constraints we emphasize that we asked stakeholders for perceptions on satisfaction and value. Perceptions are not the same as actual measurements, which is especially the case for our value measurements. We prefer to measure the real business value as delivered by each software project. However, two problems occur with regard to this.

Holistic measurements on value are often difficult to make for a single project (e.g. *return on investment* and *net present value*). Besides that, such measures (e.g. *net promotor score*) cannot easily be related to software projects, mainly because too many different factors are of influence for such measurements.

As explained in Subsection 7.5.2, two limitations are in place with regard to the setup of our electronic survey. Adopting any of the existing validated measurement instruments on customer / stakeholder satisfaction, which are available from the behavioral science, and economics and management theory, might be helpful for continuation of the survey in future research.

Secondly, the almost perfect correlations between the four aspects of *perceived value* indicate that the aspects are measuring the same construct, or that the answers to those items were influenced in the same way. We argue that it would be good to adjust the survey with regard to these aspects for future research.

7.6.2. Internal Validity

A threat to internal validity that we acknowledge is the fact that ‘fishing for p-values’ might hold a risk that some of the correlations we find are a coincidence. We limited this effect by making Bonferroni corrections for all p-values that we used in the multiple comparisons (see Table 7.8). Furthermore, the number of parameters in our model is too low to perform a reliable generalized linear model test with multiple data points. To prevent from systematic error we perform an exploratory test in which we do test for p-values, yet we confront these with findings of the qualitative analysis.

In order to minimize systematic error with regard to subjectiveness of stakeholders in their survey answers, we included representatives from both

IT and business that were involved in any way in a subject project. We considered to also include participants (from the organizations involved) that did not know the subject projects in the assessment of *perceived value*. However, the study was performed in an operational context within *BelTel* and subsequently *DutchCo*. Answering surveys, subsequent a release, was implemented as an operational capability. When designing the study we considered that it was undesirable to interfere with stakeholders more than necessary in their operational activities, and not to engage them in surveys related to projects in which they did not participate.

Another attempt we made to prevent from bias, was to perform anonymous surveys, although one can argue that based on specific roles a lack of anonymity could introduce potential bias. In order to reduce bias due to ambiguity of survey answers with regard to the four aspects of value (customer, internal process, financial, and innovation) we applied additional text on the survey that was shown when participants hovered over a question mark linked to each question.

A limitation is in place regarding the summarized themes in Table 7.9 and Table 7.10; the included key concepts are defined loosely on free text concepts provided by respondents in which they later were categorized.

7.6.3. External Validity

Concerning external validity, the extent to which the results of our study can be generalized to other companies than *BelTel* and *DutchCo* is difficult to answer because we performed multiple case studies in these two specific companies. Not all findings that occurred in the *BelTel* case were found in the *DutchCo* case too. Especially because our findings relate to specific situations, maturity, and development approaches we argue that a one-on-one generalization to other companies is not valid. Instead we argue that evidence-based software engineering (Dybå, Kitchenham, & Jorgensen, 2005) in a way we performed for this study within both *BelTel* and *DutchCo* is a precondition for mature improvement within other companies too.

7.6.4. Study Reliability

A threat related to the study's reliability lies in the fact that the lead author of this paper was a member of the measurement team within *BelTel*, and carried out the functional size measurements within *DutchCo*. However, we tried to prevent from bias by ensuring that the *BelTel* measurement team and DutchCo measurement expert are independent and objective in their collection of data. Moreover, the size measurements were made before the analyses were made and performed along the functional size measurement procedures that are repeatable independently from the actual measurement expert (IFPUG, 2009).

A link that we did not study, but that is mentioned in other studies, is the relation between *stakeholder satisfaction* and requirements (Pitangueira et al., 2016) (Maciel & Barros, 2016) and documentation (Díaz-Pace et al., 2016).

7.7. Conclusions and Future Research

The outcomes of our multiple case study indicate, that "within time and cost" does not automatically lead to satisfied stakeholders. A focus on shortening overall project duration, and good communication (e.g. no last minute surprises) and optimal collaboration between teams, has a positive effect on satisfaction of stakeholders.

On the other hand, too late delivery and long project durations, and many defects dissatisfy them. Our study does not provide any evidence that steering on costs helped to improve the satisfaction of stakeholders.

A novelty in the results of our study is that we linked *perceived value* to a set of project metrics, among others functional size of projects. As an answer to our research question –

RQ: How do stakeholder satisfaction and perceived value relate to cost, duration, defects, size and estimation accuracy of software projects?

we found the following five take-away-messages:

1. *Stakeholder satisfaction* can be improved by steering on good quality, good communication, and good quality of duration estimations. Satisfied stakeholders emphasize the delivery of good value to the customer.
2. *Stakeholder satisfaction* goes down when issues occur with testing and deployment, unclear requirements, bad documentation, hidden business rules, and requirements creep, when communication is bad, and in case of long project duration and schedule overrun.

3. *Perceived value* did not correlate with *project size*, *cost per FP*, and *days per FP*, indicating that functional size is not an indicator for value, however more research is needed to confirm this finding, since the original study indicates otherwise.
4. We identified two themes that did not apply to stakeholders or value: 'Agile' itself is not always a point of discussion in companies, even when they are agile, and *project cost* seems not an important issue for stakeholders.
5. The study also confirmed an effect known from related work: *project size* strongly correlates with the other core project metrics *project cost*, *project duration*, and *number of defects*.

Our final question is how we and others build on the main findings of this study. We see the following four aspects to be important for further research:

- How can good quality of deliverables, good communication, and reliable estimations for duration be controlled to create satisfied stakeholders?
- How can issues with testing and deployment, unclear requirements, hidden business rules, bad communication, and schedule overrun be controlled to mitigate stakeholder dissatisfaction?
- Do larger projects (measured in functional size) lead to higher *perceived value* scores?
- How can quality of project duration estimations be used to improve *stakeholder satisfaction* and *perceived value*?

7.8. Acknowledgments

We thank *BelTel*, *DutchCo*, and both *IndSup-A* and *IndSup-B* for their generosity to allow us to use company data in our research, and all survey respondents for their help on sharing their ideas on improvement of software projects with us. We thank Tableau for allowing us to use their BI solution to build our performance dashboard.

We built an approach and an accompanying tool and research repository for collecting, analyzing, and benchmarking industry data on software deliveries. We evaluated the model in case studies and surveys in industry, to demonstrate its strengths and limitations in practice.

Photo by Arnaud Mesureur

8. Conclusions

In this concluding chapter, we recapitulate the contributions of our study over time, we summarize our research questions, we discuss the findings and describe threats to validity. Finally, we describe the implications of our research for industry, research, and education, and we draw conclusions.

8.1. Contributions

The main contribution of this research can be summarized as a model for Evidence-Based Software Portfolio Management that is evaluated in four case studies with surveys included in a real-life context in industry and three data analysis studies on subsets of industry data.

Although the contribution must be assessed as a coherent set of components, four self-contained components can be recognized. The first component is (1) an approach for collecting, analyzing, and benchmarking historic industry data on software deliveries. The second (2) an accompanying EBSPM-tool for doing so in practice. The third (3) an EBSPM research repository for benchmarking purposes. Besides that, we (4) evaluated the model in a number of diverse case studies and surveys in real-life situations in an industry context. By doing so, we demonstrated the strengths and limitations of the approach in practice.

These four self-contained components are described in more detail in the following Subsections.

8.1.1. A dynamic, agile EBSPM approach

The first contribution, a part of the EBSPM-model that we built prior to the evaluation in industry, is the evidence-based approach for software portfolio management. The approach describes how EBSPM is applied in a real-life industrial context, what metrics are included in the model, and how these metrics are collected, analyzed, and benchmarked against – subsets of – other software projects in the EBSPM research repository. As our research indicates, the approach – especially where it concerns the metrics included in the model – is not a static component. The EBSPM approach is to be characterized as dynamic and agile. However, the core of EBSPM is an objectified view on the core metrics: cost, time, and quality. In our approach, we built a *cost duration matrix*, based on functional size as a normalizer. This core can be situationally extended with additional metrics.

A good example of this dynamic, agile character of the EBSPM approach is the subset of different metrics that we used to measure *perceived value*. Based on the observations related to success- and failure factors as described in Chapter 3, and mentioned by interviewed stakeholders in the Cecil-Case in Chapter 4, we developed an initial set of four metrics to measure *perceived value*. Our goal was to examine whether projects ending up in the *bad practice quadrant* of the *cost duration matrix* could be assessed by stakeholders as delivering a high amount of value. For this purpose, we defined in Chapter 7 a subset of four metrics for *perceived value*, based on Kaplan and Norton's Balanced Scorecard (Kaplan & Norton, 1995). Finally, we found that these four metrics were apparently measuring the same construct, or that answers given by stakeholders on these metrics influenced each other in a way. We assumed that measuring *perceived value* at a project or a delivery level is simply a too high, abstract level. We suggested that in future research *perceived value* should better be measured at an epic or even at a user story level. This example, in our view, clearly shows how a metric evolves over time, based on findings from evaluations of our model in an industry context.

8.1.2. An EBSPM-tool, a tool description and evaluation

With the performance dashboard and the *cost duration matrix* as a core instrument we developed an EBSPM-tool, including a tool description and a tool evaluation as described in Chapter 3. The EBSPM tool includes a performance dashboard, based on a *cost duration matrix*. The matrix divides a software company's project portfolio into four specific quadrants and recognized *bad practice* and *good practice*. Based on this we determined – as described in Chapter 4 – seven success factors for software projects, and nine failure factors.

8.1.3. An EBSPM research repository with 500 projects

The EBSPM-tool nowadays holds an EBSPM research repository with data of more than 500 finalized software projects from different business domains, different companies, and different delivery strategies. The repository is described more in detail in Chapter 3. The repository is recorded as open source at the 4TU Centre for Research Data (Huijgens, 2017a), and available for researchers and practitioners for replications and other research purposes.

8.1.4. Evaluation of the EBSPM-model in industry

Finally, an important contribution of this dissertation is to be found in the different case studies, surveys, and data analysis studies that we performed in industry to verify our findings in the quantitative analysis on our EBSPM research repository data.

In Chapter 4 we gave a description of a case study, including electronic surveys and interviews, in which a release-based, iterative process on a legacy system worked well, satisfying key stakeholders, despite a poor quality of the system itself.

In Chapter 5 we described a case study with surveys on specific use of an evidence-based approach for upfront pricing of project proposals in a globally distributed context.

In Chapter 7 we described a multiple case study, including surveys, within two different companies on the relations between *stakeholder satisfaction* and *perceived value* and software delivery performance.

In Chapter 3 and 6 we documented data analysis on subsets of historic data of software projects that we collected over time in four different software companies.

8.2. The Research Questions Revisited

As a basis for conclusions about this thesis we recapitulate our five research questions.

8.2.1. Success and Failure Factors for Software Projects

Our first research question is about factors that might influence the success or failure of software projects.

RQ1: What success factors and failure factors affect software project performance?

We found four strongly significant success factors for software projects: a steady heartbeat, fixed and experienced team, agile (Scrum), and release-based working mapped on a single application.

We identified seven failure factors for software projects: rules & regulations driven, dependencies with other systems, technology driven, once-only projects (projects that are terminated after termination, including removal of the project team), security related projects, many team changes and an inexperienced team, and migration projects.

Based on these findings we identified actions, when aiming for a *good practice* project portfolio. First, software companies should avoid *bad practice* by steering at limitation of inter-dependencies between projects and systems, stay away from unnecessary team changes, and build teams where possible with experienced team-members. To create *good practice* software companies should steer actively on organizing software development in a release-based way, set up fixed teams with experienced team-members, implement a steady heartbeat, and go for an agile (Scrum) delivery approach where applicable.

Once-only projects should be avoided whenever possible. Keeping software engineering teams together for a longer period, and more subsequent deliveries leads significantly to *good practice*, while once-only projects are linked with *bad practice*. However, when keeping teams together is not an option, companies should pay special attention to standardization and limitation of procedures to smoothen the project's progress, re-use of knowledge of other once-only projects, and implementation of a learning cycle. A good tool in such a situation is to implement the preparation of lessons learned once projects are closed as a mandatory practice.

Finally, companies should implement a long- and medium term portfolio strategy, including a learning capability, to avoid *bad practice* by limiting (where possible) projects that are characterized as technology driven, rules & regulations driven, migration of data, and security issues.

8.2.2. New Developments, Maintenance and Legacy

Our second research question addresses whether the findings about success and failure factors for software deliveries also apply to legacy applications where maintenance, enhancements, and reliability and availability are at stake, and what actions can be taken to increase the performance in such contexts.

RQ2: What actions can be taken to increase project performance when running a software project portfolio with new developments and maintenance of legacy systems involved?

For this purpose, we addressed the problem of different ways of working for evolving legacy software, which resist change. Our study confirmed three success factors that were earlier identified in research question RQ1: a steady heartbeat, a fixed and experienced team, and a release-based way or working, mapped on a single system.

Besides that, four additional success factors were identified. First, the role of the product owner and the personal interpretation of that role. Second, a focus on quick wins and small, fast deliveries of requirements based on end-user problems. Thirdly, if the role of the Scrum master is not formalized, then this leads to a self-organizing team, with an onsite lead developer that coordinates offsite team members. And finally, we found that the use of a specific product backlog management tool positively influences communication.

Converting these factors found to concrete actions, cannot be done universally, and must be completed depending on the environment. We illustrate that by means of examples we found in our research.

Based on our findings we argue that in a situation with new development in combination with enhancements on a legacy system, companies should best apply Scrum as the development approach. A light-version of Scrum, with a self-organizing team, might be a good option in such cases.

Although not deeply examined in our study, it might be a good idea to build one backlog of all user stories acting on a legacy system, instead of solving difficult performance-related requirements in separate once-only projects. The performance of a team might get worse when also such non-functional requirements are included in the backlog, but we assume that the overall performance of all projects will be better due to effects of release-based working, one experienced team, and a short feedback loop to end-users.

8.2.3. Evidence-Based Pricing of Project Proposals

Our third research question addresses whether statistics-based pricing of up-front prepared project proposals satisfies both the customer and the supplier party in a globally distributed software engineering context, and whether such an approach helps to create cost and time improvements.

RQ3: How can an empirical, evidence-based pricing approach for software engineering, be used as a single instrument (without expert judgment), to create cost transparency and cost and time improvements?

Concerning this research question, we demonstrated how an evidence-based approach can be successfully used in practice as a tool for pricing of project proposals. This method of pricing as a single instrument, without intervention of expert judgment-based opinions, leaded in the companies in our case study to an improved transparency of project proposals and satisfied stakeholders from both the customer and the supplier.

On the positive side, the applied pricing method did lead to significant cost improvements. Duration on the contrary was longer than the peer groups in our benchmark and showed a deteriorating trend, probably caused by the fact that average project size got smaller over time.

8.2.4. Cost and Effort in Measurement Repositories

Our fourth research question compares aspects of project cost and effort between our EBSPM research repository and a commonly used benchmark repository from the International Software Benchmark Standards Group (ISBSG).

RQ4: How do the EBSPM-repository and the ISBSG-repository compare on size, cost, effort, duration and number of defects, and how can differences be explained?

We compared two industrial, yet publicly available software project repositories, the EBSPM-repository and a subset of the ISBSG-repository, to analyze differences regarding *project cost*, *project size*, *project duration*, and *number of defects*. We determined suitability of some key variables (*size*, *duration* and *number of defects*) of both data sets for cost prediction.

We demonstrated a (log)-linear relation between *project cost* on the one hand, and *project size*, *project duration* and *number of defects* on the other. This justifies conducting linear regression for cost. Besides that, we established that ISBSG is substantially different from, e.g., EBSPM, in terms of *project cost* (cheaper) and *project duration* (faster), and the relation between *cost* and *effort*. This implies that practitioners and researchers alike should be cautious when drawing conclusions from a single repository.

Finally, we showed that while in ISBSG effort is the most important cost factor, this is not the case in other repositories, such as EBSPM in which size is the dominant factor.

Based on our findings we argue that, supported by the importance of both effort and cost data for decision makers in industry, effort and cost should be treated as different metrics in research.

8.2.5. Stakeholder Satisfaction and Perceived Value

For our fifth research question, we included two qualitative aspects, *stakeholder satisfaction* and *perceived value*, into the equation.

RQ5: How do stakeholder satisfaction and perceived value relate to software project performance?

Outcomes of our multiple case study indicate that “within time and cost” does not automatically lead to satisfied stakeholders. A focus on shortening overall project duration, and good communication and optimal collaboration between teams, has a distinct positive effect on satisfaction of stakeholders. On the other hand, too late delivery and long project durations, and many defects dissatisfy them. Our study does not provide any evidence that steering on costs helped to improve the satisfaction of stakeholders. A novelty in the results of our study is that we linked *perceived value* to a set of project metrics, among others functional size of projects.

8.3. Discussion

Interest in the field of software analytics has grown rapidly in both the research and business sectors (Menzies & Zimmermann, 2013). This dissertation addresses steering software project portfolios based on analysis of such data. Yet, where we struggled with the manual collection of software delivery data and challenges in acceptance by stakeholders due to this, nowadays many software companies tend to move towards automated pipelines with toolsets that log data continuously. Automated mining and analysis of data within the log files of these tools quickly becomes a reality. Therefore, we expect these developments to have a positive impact on the further growth of software analytics as a topic in research and in industry.

Although the EBSPM-model as described in this thesis is built on manual collection of software deliveries, we think that further development in the direction of an automated way of collecting, analyzing, and visualizing software portfolio metrics is a logical next step. One of the main lessons learned from our study is that software companies should collect their own data of finalized deliveries to analyze effects that are typically for their own organization, instead of relying on repositories with data of other companies. Furthermore, the study gave us insight in the backgrounds of success and failure within software project portfolios, and the relations between core metrics such as *time*, *cost*, and *quality* on the one hand and *stakeholder satisfaction* and *perceived value* on the other.

Yet, at the same time there are many things we do not know about software portfolio management. One aspect stands out when overlooking the results of our study and reflecting with progress in research and in industry: the aspect of value (measuring the benefits of software deliveries). RQ5 relates to how *perceived value* relates to software project performance. For this purpose, we studied the effects of *perceived value* on *functional size*, *cost*, *time*, *quality*, and *stakeholder satisfaction*. One of the things that we were curious about was whether we would find correlations between *functional size* and *perceived value*. This would confirm that the EBSPM model – since this is based at *functional size* as a normalizer – would be a good predictor of deliveries with high benefits in terms of *perceived value*.

Although we found promising, but somewhat weak correlations between *perceived value* and *stakeholder satisfaction* on the one hand, and *time*, *cost*, and *quality* on the other, we assume that especially *perceived value* needs to be measured at a lower level, such as *user story* or *epic* to find better results in correlation tests. However, a limitation is in place here. Value is influenced by speed and attention too, and in our case in a way a subjective measure. Therefore, maybe other factors might influence value perception when measured at a lower level too.

8.4. Threats to Validity

Construct Validity

Regarding the degree to which a test measures what it claims to be measuring a remark is in place on the manual collection of data, and especially on function point analysis (FPA). However, we tried to mitigate risk regarding collected data by building in several checks and balances in our approach. All project data was reviewed with the applicable project manager, with applicable release manager(s), and with (business) executives. All cost related data was also reviewed by the financial controllers of the applicable software companies.

We used functional documentation as a source for FPA; a consequence is that low quality documentation could have led to low quality FPAs, however, we thoroughly reviewed all sets on completeness and correctness.

The fact that we used electronic surveys to ask stakeholders of software deliveries for perceptions on satisfaction and value, can be looked upon as a threat to construct validity. We realize that perceptions are different from real measurements, and we certainly prefer to measure real business value. However, as described earlier holistic measurements on value are often difficult to make for a single delivery, and such measures cannot easily be related to individual software deliveries, due to complexity of influencing factors. We tried to mitigate the effect of subjectivity of stakeholders by including representatives from both IT and business in our surveys, and by implementing the surveys as an operational capability. All surveys were performed in an anonymous way.

Internal Validity

By normalizing all project data with the functional size in FPs we warranted internal validity, the extent to which a causal conclusion is based on our study. This enabled objective comparison of delivery performances and minimizes systematic error.

A threat to internal validity that we acknowledge is the fact that ‘fishing for p-values’ might hold a risk that some of the correlations we found are a coincidence. We limited this effect by making statistical corrections were applicable.

In the collection of data of finalized projects, we asked project managers, Scrum-masters, or product owners to provide us with relevant project or delivery data. We also asked stakeholders to provide us with the applicable delivery approach, where we used ‘plan-driven’ for projects that were reported to be performed in a waterfall kind of way. Deliveries that were performed in an agile way were marked as ‘Scrum’ in our repository. We realize that many hybrid forms or lighter forms of Scrum could be applicable too. The actual implementation of the delivery methodology is not studied in detail in our research.

External Validity

Concerning external validity, the extent to which the results of our study can be generalized to other companies than assessed in our study, we argue that our different case studies and their findings relate to specific situations, maturity, and development approaches in different software companies, and therefore cannot be generalized without much reluctance to other companies.

Also, findings in our study on huge differences between software project repositories indicate that generalization of our findings is not a good approach for software companies that want to use our results. Our best advice would be to start collecting and analyzing data of own finalized software deliveries, by making use of our collection and analysis methods.

Study Reliability

Finally, a threat related to the study’s reliability lies in the fact that the author of this thesis was personally involved as a software analytics consultant in all case studies and data analysis activities.

However, we tried to avoid as much bias as possible by objectively looking at the collected deliveries and by ensuring that all data was peer reviewed by other members of the measurement teams and by stakeholders such as financial controllers, project managers, and release managers too. Peer review of collected data of finalized projects was in all case studies implemented as a mandatory quality assurance step in the measurement and analysis process. The involvement as a consultant in practice might even be a prerequisite for obtaining this kind of data, since it must provide value for the companies concerned.

All data that we used within the scope of this thesis for quantitative analysis is available for researchers and practitioners through the 4TU Centre for Research Data (Huijgens, 2017a).

8.5. A reflection on the empirical methods used

Looking back over a period of four years of research on software project portfolio performance improvements in practice, we feel it is valuable to discuss the experiences with the research methods used by us. As explained in the introduction on our research method and the evaluation in industry practice in Chapter 1, we made use of an empirical approach, where we proposed a model, that we subsequently evaluated through empirical studies in an industrial context, in our case with case studies and surveys. Besides that, we performed three data analysis studies on the historic dataset collected in the EBSPM research repository.

8.5.1. Case Studies

We performed four case studies in typical industry settings. Three of them being single case studies, one was a multiple case study that was performed in two different companies. We performed all case studies in an iterative way, meaning that we designed each study as a logical follow-up of earlier performed ones. Although our main focus from the start was on ‘how to measure value and satisfaction?’, the requirements of each separate study were largely determined by the practical context. We designed all case studies according to theory on case studies (Wohlin, et al., 2000) (Yin, 2008) (Runeson et al., 2012). Yet, a challenge was that in the setup and conducting of the case studies we were driven by continuously changing strategies within the companies where the studies were conducted. This forced us to opt for a flexible setup of our case studies. In a way, one might argue that we performed some form of action research instead of case studies by-the-book.

We would like to emphasize that this phrase "changing strategies" might seem to imply being-out-of-control or indecisiveness, but that was not the case. The applicable companies where continuously in ongoing transformations of their software delivery teams. One company was transforming towards a new, long-term strategic alliance with an Indian external supplier. All applicable companies were amid implementing new ways of working, often agile (Scrum) or DevOps. And all companies had clearly made a conscious choice to organize their software delivery in an agile and iterative way. As the Agile Manifesto (Beck, et al., 2001) says it; they were responding to change over following a plan.

Based on our observations in practice, we assume it would be useful for the software engineering research community to further study the backgrounds of conducting case studies in a practical situation with a continuously changing context. In a way this is already observed by (Runeson et al., 2012), where they refer to (Benbasat et al., 1987) mentioning the *lack of experimental control*. We feel that additional guidelines for researchers in practice - applying to modern development approaches - would be helpful. We believe that alike development of software solutions in industry, the pursuit of a steady heartbeat, two-weekly sprints, prioritizing a backlog, and organizing demos and retrospectives, can contribute significantly to a greater probability of successful case studies performed in industry.

8.5.2. Surveys, electronic questionnaires, and interviews

Within the scope of our research we performed electronic surveys in three studies. In one study, the electronic surveys were completed with a series of structured interviews with project stakeholders. Although we expected to encounter resistance among the project stakeholders when filling in electronic questionnaires, this has not been the case in practice.

We assume that the fact that we implemented the electronic questionnaires from the start as an operational practice right after finalizing a release, helped a lot here. People that were involved in a release – such as software developers, testers, release managers, project managers, product owners, and Scrum-masters - soon became accustomed to receiving a questionnaire for the completion of a project for each project that was released to production.

However, a trend that we observed was that as software organizations grow more into automation and continuous delivery, the amount of surveys increases drastically. Mature software companies tend to implement continuous experiments, especially regarding customer experience in frontend environments such as Internet and mobile apps.

Based on our observations in practice we assume that software companies more and more need to find a balance between mining their continuous delivery systems for data, instead of surveying stakeholders in delivery teams and collecting qualitative data through electronic surveys, to prevent *survey fatigue*.

We think that a final remark regarding structured interviews is in place. In one of the case studies - described in Chapter 5 - we have also conducted interviews with involved stakeholders in a series of releases on a legacy application in addition to electronic questionnaires. We observed that especially conducting interviews with directly involved can greatly contribute to a better understanding of the backgrounds of software deliveries. This is underlined by the fact that the particular case study is the only one in this thesis that has been given a specific name that is being cherished by all the parties involved: *the Cecil-case*.

8.5.3. Data Analysis Studies

Within the scope of our research three data analysis studies were performed. In the first analysis study – captured in Chapter 4 – we analyzed our initial subset of finalized software projects from the EBSPM research repository for success- and failure factors. In this study, we did not formally design the collection of all data upfront, resulting in the fact that we had to do the analysis with the set of metrics as collected. In the second analysis, we compared the projects in our EBSPM research data set with a subset of projects from the ISBSG repository. In both studies data quality played a decisive role. Mapping both studies on a formal standard for software metrics, such as ISO/IEC 25023 (Nakai et al., 2016), might help to solve data quality issues in future.

8.6. Implications

In this concluding section, we describe the implications of our findings for successively three aspects of our research practice: research, industry, and education.

8.6.1. Implications for Research

Outlining a look at follow-up research in the near future in the field of software economics, we elaborate a little deeper on four emerging aspects in software economics research that we believe to be of great importance for valorization to the software industry.

Software analytics with industry involvement

In ongoing work that we perform in close cooperation with ING Bank, we examine whether our model can be applied in a continuous delivery environment in a software company with worldwide more than 300 teams, that performs more than 2500 deployments to production each month on more than 750 different applications (Huijgens et al., 2017b). We performed an exploratory case study that focuses on the classification based on predictive power of software metrics, in which we analyzed log data derived from two initial sources within this pipeline.

If this exploratory study explains something unequivocally, it is that automation is an undeniable necessity for software analytics. The scale of teams delivering ongoing software solutions becomes that large, that manual collection of data – as we did within the scope of this thesis – is not a futureproof option for many software companies. Based on the outcomes of our exploratory study, we argue that mining data from the logs of tools within the delivery pipelines, combined with advanced statistical analysis of the log data, to identify metrics that correlate strongly might be an interesting way forward for software analytics.

Based on the relative low amount of related work about automation of software analytics we have the impression that the research community lags the developments in industry about continuous delivery and automation of pipelines and the use of software analytics to support decision-making. Where many software companies are already implementing solutions for automation, or start to think about these, researchers seem to have difficulties setting up collaborations with industry on data collecting for research purposes.

We regard this disadvantage in research communities in this area as a major threat to future research into the backgrounds of continuous delivery and automation of software development pipelines. Subjects such as software analytics in a continuous delivery context should be on top of the agenda of contemporary software engineering researchers, to eliminate the backlog that is currently there.

Automation of dynamic dashboards for decision-making

Where in the example in the former Subsection the aspect of automation and building a performance dashboard itself is not within the scope of the analysis, we expect that automation is one of the main requirements in future research and technical solutions that focus on enabling software analytics in software companies.

The ultimate goal of our analysis is to explore *good practice* and *bad practice*, as we described in Chapter 3 of this thesis; e.g. 'good' deliveries (being better than average within the scope of a software delivery team) and 'bad' deliveries (being worse than average within the scope of a team) (Huijgens et al., 2014c). By doing so we expect to identify success factors that help software delivery teams to create better deliveries in future releases, and failure factors that help teams to prevent from 'bad' deliveries.

Driven by the ongoing objective within software companies to automate the delivery pipeline, we expect that automation will also be a major requirement for software analytics components within such pipelines. An emerging challenge for software measurement is the fact that software analytics solutions should be as far as possible automated too, to be successfully applicable in industry.

Functional Size Measurement threatens to disappear from the metrics palette

An additional challenge as a result from this strive towards automation might be the quest for a useful, and powerful normalizer for software deliveries. In this thesis, we use *functional size measurement* (FSM), in our case IFPUG Function Points (IFPUG, 2009), to compare performances of different software deliveries. However, we expect that due to the threat of automation FSM threatens to disappear from the software metrics palette due to its manual processing, and due to its somewhat old-fashioned and subjective imago among software engineering stakeholders.

Functional size in its actual form, a commonly used industry standard that measures the scope of software deliveries, and that we assess in the EBSPM approach to be a significantly strong metric with excellent properties for normalizing software deliveries, simply does not fit in a highly automated continuous delivery pipeline approach. Yet, at the same time the strongest correlations are to be found between size related measurements and cost, duration, and number of defects, as we showed in our research (Huijgens et al., 2014c) (Huijgens et al., 2017d).

A potentially interesting solution to this FSM problem might be compression to measure the amount of new functionality or changes in software quality as demonstrated by (Raemaekers et al., 2015).

We argue that automation of FSM in whatever form, is unmistakable needed in software companies that mature in an agile way of working. Remarkably enough FSM stakeholders from industry and research do recognize the need for automation. A vast majority (87%) of 336 FSM stakeholders that answered a survey that we performed in 2016 on the importance of automated FSM based on code (Huijgens et al., 2015b) considers FSM to be an important tool for decision-making. 42% Of the respondents says automated FSM based on code is important, although many are uncertain (neutral) about this. A clear majority of respondents (50%) expects it to be difficult, while many are neutral on the question whether this idea will be difficult.

Based on the survey outcomes, we speculate that a solution for automated FSM that focusses on program code can help both FSM experts and decision makers. Such a solution can help in agile and highly automated delivery environments. Due to the assumed difficulties of automation of FSM based on code – the difference between a functional and a technical view, and the diversity in programming languages – we think that a focus within the research community on translation from technical programming code towards functional counting rules might be of importance. We assume that close cooperation with FSM communities on this topic will be valuable for translation towards industry (Huijgens et al., 2015b).

Integrating value and stakeholder satisfaction

In its ultimate form, the ideas as described above can lead to a fully automated software analytics solution that can be implemented in a software company's continuous delivery pipeline. In such a scenario, a model alike the *cost duration matrix* might form the core element of a dynamic performance dashboard. Based on ongoing analysis of log data of the tools from this pipeline, the dashboard focuses on strong metrics; metrics that have strong prediction power on to be started and ongoing software deliveries.

Yet, compared to the research as described in this thesis one important set of metrics is still missing in such a scenario. Qualitative metrics such as *stakeholder satisfaction* and *perceived value* are not stored in any of the tools within the pipeline, and therefore are not included in the analysis underneath the automated dashboard. Driven by the importance of such qualitative metrics to understand the causes behind a software company's performance as depicted in quantitative metrics, we argue that an automated solution should include a subset of both *quantitative* and *qualitative* metrics. There is a need for non-process metrics that are not in scope of automated process tools, and that should be collected through qualitative analysis such as electronic surveys or interviews and text analysis.

As the ultimate goal, we imagine a fully automated software analytics solution that can easily be 'plugged in' into a software company's continuous delivery pipeline, combined with regular electronic surveys and interview sessions to understand the causes behind a company's performance as depicted in quantitative performance indicators. To get there, substantial additional research is required on ways to automatically create performance dashboards that have sufficient learning ability to help software companies to realize actual innovation. We need to better understand what metrics really help to create improvement, and what metrics can be ignored, and how qualitative metrics can help us to understand the reasoning behind change.

8.6.2. Implications for Industry

Although we positioned EBSPM, alike EBSE, as an empirical study in the field of software economics, we felt from the start that especially in the software industry some challenges were applicable.

Data Collection Issues

A first hurdle that we had to take had to do with the way data was collected. At the start of the study, we already had access to a dataset of 352 completed projects from three large software organizations. While this ensured that we could quickly start with the analysis of the project data, and we had to spent relatively little time on data collection, this meant that we had to do with the information we had at that time. For most of the projects in the dataset no additional data could be collected, since the projects were finalized and discharged.

During the study, we added more software projects to this initial dataset, building up to a research repository of more than 500 finalized software projects in four different companies.

A major drawback regarding the practical method of data collection was, that we regularly had to deal with challenges related to data quality. An example of such issues is the limited availability of effort data, especially in globally distributed environments. Due to this, we decided to focus our study on cost data, instead of effort data. The main rationale for this decision was, that we experienced that decision makers in practice generally seem to base their decisions on costs, and to a much lesser extent on effort.

Preliminary results with collecting data from the logs of tools in the automated pipeline in a continuous delivery organization taught us that automation as such is not going to solve these data quality issues. Silo-behavior by team members that 'own' tools in a pipeline, and badly linked data between different tools in a pipeline cause new challenges for software analytics. Yet, we assume that automation makes that once such problems are solved, a mature solution can set a standard for future use.

Huge diversity in development aspects

Because of the emphasis in our research on the entire software project portfolio of organizations, diversity in our research repository was huge. The dataset held data from once-only projects to release-based deliveries, from delivery approaches such as plan-driven (waterfall) to agile (Scrum), with many hybrid in-betweens, from in-house performed projects to all varieties of sourcing (offshore, outsourcing, globally distributed development teams), a mix of business domains (e.g. internet and mobile apps, data warehouse and business intelligence solutions, legacy payment systems, client management applications), and a huge variety of programming languages. An important principle of our study was to develop an approach that could handle this large variety of development issues.

An important implication for industry regarding this observation is, that it makes sense for software companies to consider these when planning new projects. Estimation practices can be highly influenced by differences in business domains and delivery approaches. Companies would benefit when they collect historic data from their own projects, instead of relying on external benchmark sources.

Driven by application in industry

A third challenge is about valorization. The practical application of the results of the study into the daily practice of software organizations that participated, largely determined the chosen approaches and methods. To address this challenge, we organized the study in an industry-oriented way: where possible we opted for a case study as the appropriate method to perform research. An important reason to do so was the fact that the author of this thesis performed the Ph.D. besides his daily job as a software analytics consultant. For all case studies that are included in our research we worked in close cooperation with software companies that hired the author to help them improve their software delivery processes.

An aspect that we did not study in this thesis, but that might play an important role in the success of our approach, is the extent to which our empirical findings can be disseminated in industry. Practitioners are positive towards software engineering research (Lo, Nagappan, & Zimmermann, 2015). Yet, at the same time they have very strong beliefs that are primarily formed based on personal experience, rather than on findings in empirical research, and that these beliefs do not necessarily correspond with actual evidence (Devanbu, Zimmermann, & Bird, 2016). Maybe aspects of game development can play an important role in future research on this topic (Murphy-Hill, Zimmerman & Nagappan, 2014).

This implies that software companies can benefit from setting up research in a way that fits their context. For example software companies could support employees to perform a part-time Ph.D. besides their daily job. Another example is to incorporate staff in ongoing research activities, so that research fits more closely with the daily practice of software engineers.

Steering on 'within time and budget' does not help to really improve

The outcomes of our multiple case study indicate, that "within time and cost" does not automatically lead to satisfied stakeholders. Based on the outcomes of our study we argue that executives that define project success as 'finishing within budget and time' are blinkered, and therefore neglect to really improve their companies' performance.

8.6.3. Implications for Education

Looking at the implications of our findings for education, we argue that especially the trends on automation of delivery pipelines, and the focus at building automated solutions for software analytics should be addressed to students in the domain of software engineering too. As in industry and research, software analytics also seems to be lagging as a subject of education. The first focus in education is at the 'core' subjects of software engineering – design, build, test, deploy, architecture – and software analytics comes in scope once these are matured. A risk that comes with this approach is that setting up proper education about software analytics comes too late for industry to be of real use, and specialists in industry will setup their own education channels for this purpose.

Furthermore, we assume that the success- and failure factors that we identified in our research should be taught to students.

8.7. Conclusions

Looking back at our research period the question can be raised 'What do we know now, what we not yet knew when we started our study?'

We developed an approach, including an accompanying tool that can successfully be used in industry to benchmark the performance of a company's software portfolio and that helps analyzing a company's performance in terms of *time*, *cost*, *quality*, *stakeholder satisfaction*, and *perceived value*. The data that we collected in our repository gave us a deeper insight into the factors that are related with success and failure of software projects, from a portfolio management point of view.

We learned, among others by comparing the characteristics of our repository with other large repositories that it is wise for software companies to collect and analyze their own historic software portfolio data, because cross-company large differences in performance are found. We obtained a better understanding of the differences and equalities between effort and cost of software deliveries. Additionally, we studied the effects of pricing of software deliveries, giving us a better insight into ways to support decision-making.

Based on the results of ongoing research, we expect that automation of the measurement and analysis process, based on statistics to calculate strong relationships, is a direction in which the analysis of software portfolio (software analytics) is the to develop strongly in the coming years.

List of Abbreviations

AFP *Automated Function Points*, a standard based on the IFPUG method (Object Management Group (OMG), 2014) that was developed by the Object Management Group (OMG).

COSMIC *Common Software Measurement International Consortium*, started in 1998, that developed an advanced, open-source method of measuring a functional size of software, as defined in ISO/ IEC 19761:2011: COSMIC FSM method (COSMIC, 2011).

EBSE *Evidence-Based Software Engineering*, an approach concerned with determining what works, when and where, in terms of software engineering practice, tools and standards, inspired by the evidence-based paradigm as employed in clinical medicine, adapting the evidence-based practices to meet the rather different characteristics of Software Engineering, and the consequences that these characteristics have for empirical studies (Kitchenham, Dybå, & Jørgensen, 2004).

EBSPM *Evidence-Based Software Portfolio Management*.

ISBSG *International Software Benchmarking Standards Group*: a not-for profit organization, founded in 1997 by a group of national software metrics associations, that aims to promote the use of IT industry data to improve software processes and products.

IFPUG *International Function Point User Group*, a non-profit, member-governed organization founded in 1986, that owns Function Point Analysis (FPA) as defined in ISO standard 20296:2009 which specifies the definitions, rules and steps for applying the IFPUG's functional size measurement (FSM) method (IFPUG, 2009).

FP *Function Point*, a unit of measurement that expresses the quantity of user functionality of an information system (as a product or as a project). Function points are used to compute a functional size measurement (FSM) of software.

FPA *Function Point Analysis*, designed by Albrecht in 1979 (Albrecht, 1979) to estimate size of software delivery by means of user functionality.

FSM *Functional Size Measurement*, is an industry standard to measure size of software engineering activities. With ISO/ IEC 14143 as an umbrella standard, five FSM methods are certified by ISO as an international standard:

1. ISO/IEC 19761:2011: COSMIC FSM method (COSMIC, 2011);
2. ISO/IEC 20926:2009: IFPUG FSM method (IFPUG, 2009);
3. ISO/IEC 20968:2002: MkII FPA FSM method (UKSMA, 2002);
4. ISO/IEC 24570:2005: Nesma FSM method version 2.1 (Nesma, 2005);
5. ISO/IEC 29881:2010: FiSMA FSM method version 1.1 (FiSMA, 2010).

Nesma *Netherlands Software Metrics user Association*, a non-profit, member-governed Netherland's-based organization founded in 1989, that owns Function Point Analysis (FPA) as defined in ISO standard 24570:2005: Nesma FSM method version 2.1 (Nesma, 2005).

OMG *Object Management Group*, an international, open membership, not-for-profit technology standards consortium, founded in 1989, that developed a standard on Automated Function Points (AFP) based on the IFPUG method (Object Management Group (OMG), 2014).

SLR *Structured Literature Review*, a tool developed by Kitchenham et al. (Kitchenham, Dybå, & Jørgensen, 2004) within their EBSE-approach, to inventory research studies according to an explicit and reproducible method.

SP *Story Point*, a roughly estimate by experience and analogue of the relative 'size and complexity' of each user story compared to other user stories in the same project. Story Points are used to determine velocity of Scrum-teams.

Samenvatting

Softwarebedrijven, zoals banken, verzekeraars, telecombedrijven en overheidsorganisaties, geven jaarlijks tientallen tot soms honderden miljoenen euro's uit aan het maken van nieuwe software en onderhouden van bestaande (*legacy*) systemen. Daarbij lopen die bedrijven tegen een aantal problemen op. Zo is het niet gemakkelijk om te bepalen welke softwareprojecten nu succesvol zijn en welke juist niet, en wat daarvan dan de redenen zijn. Dit speelt met name een rol in grote en hybride software portfolio's waarin naast nieuwgebouwde applicaties vaak ook oude en complexe legacy systemen voorkomen.

Een ander probleem is dat veelal naar bestede uren (*effort*) en kosten wordt gekeken alsof die equivalent aan elkaar zijn. Veel onderzoekers stellen dat kosten van softwareprojecten eenvoudig af te leiden zijn van de bestede uren. Maar in de praktijk blijkt dit niet waar te zijn, en zeker wanneer je kijkt naar de prijs die betaald wordt voor softwareprojecten, dan zie je dat die vaak niet transparant is en niet gebaseerd op een deugdelijke onderbouwing.

Kosten, doorlooptijd en kwaliteit – ook wel de *core metrics* genoemd – zeggen niet alles. We weten niet goed hoe die *core metrics* zich verhouden tot tevredenheid (*stakeholder satisfaction*) en de opgeleverde waarde van projecten. Zijn projecten die relatief weinig kosten en snel worden opgeleverd ook waardevol? Of ligt dat toch ingewikkelder?

En een laatste probleem gaat over de toenemende behoefte in het bedrijfsleven aan manieren om te meten en te analyseren die de toenemende graad van automatisering en *software analytics* ondersteunen.

Als een antwoord op die problemen hebben we in het kader van ons onderzoek een model ontwikkeld; *Evidence-Based Software Portfolio Management* (EBSPM). EBSPM is gericht op het helpen van softwarebedrijven om meer en beter inzicht te krijgen in de prestaties van software engineering projecten in hun portfolio. De aanpak bestaat uit drie onderdelen. Het eerste onderdeel is een beschrijving van de manier waarop we data verzamelen, analyseren en vergelijken met andere soortgelijke projecten. Het tweede is een dataset waarin we alle informatie die we verzamelen van afgeronde softwareprojecten op een eenduidige manier vastleggen; de *EBSPM research repository*. En het laatste onderdeel is een *performance dashboard* waarin we de prestaties van een softwarebedrijf of een subset van projecten vergelijken (*benchmarken*) met die van anderen.

Omdat we ons onderzoek hebben uitgevoerd in nauwe samenwerking met software bedrijven hebben we ervoor gekozen om voor dat onderzoek wetenschappelijke methodes te kiezen die zoveel mogelijk aansluiten bij die nauwe band tussen wetenschap en praktijk. In dit proefschrift zijn hoofdstuk 3 tot en met 9 ongewijzigde weergaven van al eerder gepubliceerde wetenschappelijke artikelen. En een groot deel van die publicaties is gebaseerd op een praktijkstudie (*case study)* in combinatie met elektronische vragenlijsten (*surveys*) en in één geval ook gestructureerde interviews.

Eén van de onderzoeksvragen die we in ons onderzoek stelden gaat over factoren die succes en falen van softwareprojecten in een portfolio kunnen verklaren. We vonden vier factoren die in sterke mate bepalend waren voor succesvolle softwareprojecten; een vast terugkerend oplevermoment (*steady heartbeat*), een releasematige manier van werken, agile (Scrum) als ontwikkelmethode en een vast en ervaren team. We vonden daarnaast ook factoren die daarentegen juist kenmerkend waren voor slecht presterende softwareprojecten, zoals projecten voor wet- en regelgeving, afhankelijkheden van andere systemen of projecten, projecten waarin gebruik wordt gemaakt van nieuwe technologie en teams met veel wijzigingen en veel onervaren teamleden.

Verder hebben we gekeken naar de verschillen tussen de bestede uren in softwareprojecten (*effort*) en de kosten daarvan. We ontdekten dat bestede uren en kosten weliswaar aan elkaar gerelateerd zijn, maar dat die relatie complex is en dat kosten niet altijd eenduidig kunnen worden berekend op basis van bestede uren alleen. We vonden grote verschillen tussen de prestaties in termen van tijd, geld en kwaliteit van verschillende organisaties én tussen verschillende business domeinen. We adviseren organisaties daarom altijd hun eigen historische gegevens van afgeronde softwareprojecten te verzamelen en niet zondermeer te vertrouwen op gegevens van andere bedrijven of openbaar beschikbare onderzoeksgegevensbestanden.

Naast de prestatiekenmerken als tijd, geld en kwaliteit hebben wij onderzocht wat de effecten zijn van tevredenheid van mensen die bij een softwareproject betrokken zijn, zoals projectleiders, teamleden en release managers, en de toegevoegde waarde die een project oplevert. Hiervoor ontwikkelden wij twee specifieke metrieken; *stakeholder satisfaction* en *perceived value*. Ons onderzoek laat zien dat projecten die binnen budget en op tijd (volgens de geplande kosten en opleverdatum) presteren niet automatisch leiden tot tevreden stakeholders. Vooral een focus op een korte doorlooptijd, goede communicatie en optimale samenwerking binnen een project leidt tot meer tevreden betrokkenen. Later dan gepland opleveren, een lange doorlooptijd en veel fouten zijn factoren die juist leiden tot ontevreden betrokkenen.

Op basis van de uitkomsten van lopend onderzoek, verwachten wij dat automatiseren van het meet- en analyseproces waarbij op basis van statistiek dergelijke sterke relaties worden berekend een richting is waarin het analyseren van software portfolio's (*software analytics*) zich de komende jaren sterk zal ontwikkelen.

Curriculum Vitae

Hennie Huijgens was born on November 11th 1957 in Alphen aan den Rijn, The Netherlands. He lives in Amsterdam, The Netherlands, is married and has two children.

Education

2013 - 2018	(Part-time) PhD at Delft University of Technology (TU Delft); Evidence-Based Software Portfolio Management.
2008 - 2011	MSc in Information Management, University of Amsterdam.
1984 - 1986	Academy of Art in Amsterdam, the Netherlands (1e degree teaching art and art history).
1979 - 1984	Teacher education VL/VU in Amsterdam, the Netherlands (2e degree teaching art).
1975 - 1979	Graphical Technical School in Amsterdam, the Netherlands.

Working Experience

2007 - present	Goverdson (own company); Software Analytics Specialist.
2005 - 2007	Amsterdam University of Applied Science; Senior Account Manager IT & Facilities.
2002 - 2005	PinkRoccade Atribit; Project Manager / Senior Consultant.
1998 - 2002	PinkRoccade Finance; IT Consultant.
1993 - 1998	PinkRoccade Finance; IT Professional.
1987 - 1993	Fokker Aircraft; IT Professional.
1986 - 1987	Fokker Aircraft; Console Operator.

References

Abran, A., Dumke, R., Desharnais, J., Ndyaje, I., & Kolbe, C. (2002a). A strategy for a credible & auditable estimation process using the ISBSG International Data Repository. *IWSM*.

Abran, A., Silva, I., & Primera, L. (2002b). Field studies using functional size measurement in building estimation models for software maintenance. *Journal of Software Maintenenance and Evolution: Research and Practice, 14* (John Wiley & Sons, Ltd.), 31-64.

Abran, A., Desharnais, J.-M., Zarour, M., & Demırörs, O. (2014). Productivity based software estimation model: an economics perspective and an empirical study. *9th International Conference on Software Engineering Advances–ICSEA*.

Agarwal, N., & Rathod, U. (2006). Defining 'success' for software projects: An exploratory revelation. *International journal of project management, 24*(4), 358-370.

Albrecht, A. (1979). Measuring Application Development Productivity. *Joint Share Guide, and IBM Application Development Symposium 14-17 October 1979*, (pp. P.83-92). Monterey, California.

Andreesen, M. Why Software is Eating the World. *The Wall Street Journal*. August 20, 2011.

Angelis, L., Stamelos, I., & Morisio, M. (2001). Building a Software Cost Estimation Model Based on Categorical Data. *IEEE Seventh International Software Metrics Symposium (METRICS)*. London, UK.

Arnold, M., & Braithwaite, T. (2015, June 18). Banks' ageing IT systems buckle under strain. *Financial Times*.

Atkinson, M., Sinha , A., Hass, S., Colman, S., Kumar, R., Brod, M., & Rowland, C. (2004). Validation of a general measure of treatment satisfaction, the Treatment Satisfaction Questionnaire for Medication (TSQM), using a national panel study of chronic disease. *Health and Quality of Life Outcomes, 2*(12). doi:DOI: 10.1186/1477-7525-2-12

Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International journal of project management, 17*(6), 337-342.

Auquier, P., Pernoud, N., Bruder, N., Simeoni, M., Auffray, J., Colavolpe, C., & Sapin, C. (2005). Development and Validation of a Perioperative Satisfaction Questionnaire. *Clinical Science, Anesthesiology 6*(102), 1116-1123.

Bala, A., & Abran, A. (2016). Use of the Multiple Imputation Strategy to Deal with Missing Data in the ISBSG Repository. *Journal of Information Technology & Software Engineering, 6*(1), 1-12.

Barki, H., Rivard, S., & Talbot, J. (1993). Toward an assessment of software development risk. *Journal of Management Information Systems, 10*, 203-223.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., & Kern, J. (2001). *Manifesto for Agile Software Development*. Retrieved from www.agilemanifesto.org

Benestad, H., & Hannay, J. (2011). A comparison of model-based and judgment-based release planning in incremental software projects. *Proceedings of the 33rd International Conference on Software Engineering. ACM*.

Bensabat, I., Goldstein, D., & Mead, M. (1987). The case research strategy in studies of information systems. *MIS Q, 11*(3). doi:10.2307/248684

Bergeron, F., & St-Arnaud, J.-Y. (1992). Estimation of information systems development efforts: a pilot study. *Information & Management, 22*(Elsevier), 239-254.

Bhardwaj, M., & Rana, A. (2016). Key Software Metrics and its Impact on each other for Software Development Projects. *ACM SIGSOFT Software Engineering Notes, 41*(1), 1-4.

Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., & Grünbacher, P. (2006). *Value-Based Software Engineering*. Berlin Heidelberg: Springer.

Boehm, B., (1984). Software Engineering Economics. *IEEE Transactions on Software Engineering, 10*(1), 7-19.

Boehm, B., Abts, C., & Chulani, S. (2000a). Software development cost estimation approaches - A Survey. *Annals of Software Engineering, 10*, 177-205.

Boehm, B., & Sullivan, K. (2000b). Software Economics: A Roadmap. *ACM Future of Sofware Engineering*. Limerick, Ireland.

Boehm, B., (2003). Value-Based Software Engineering. *ACM SIGSOFT Software Engineering Notes, 28*(2), 1-12.

Boehm, B., (2006a). A View of 20th and 21st Century Software Engineering. *IEEE International Conference on Software Engineering (ICSE)*. Shanghai, Ghina.

Boehm, B., (2006b). Some future trends and implications for systems and software engineering processes. *Systems Engineering, 9*(1), 1-19.

Bollen, K., & Jackman, R. (1990). Regression Diagnostics: An Expository Treatment of Outliers and Influential Cases. In J. Fox, & J. Long, *Modern Methods of Data Analysis* (ISBN 0-8039-3366-5 ed., pp. 257–91). Newbury Park, CA: Sage.

Briand, L., Langley, T., & Wieczorek, I. (2000). A replicated assessment and comparison of common software cost modeling techniques. *ACM Proceedings of the 22nd International Conference on Software Engineering*.

Bryde, D. (2005). Methods for managing different perspectives of project success. *British Journal of Management, 16*(2), 119-131.

Buglione, L., & Ebert, C. (2011). Estimation tools and techniques. *IEEE Software, 28*(3), 91-94.

Charette, R. N. (2005). Why Software Fails. *IEEE Computing / Software, September*, 1-9.

Cheikhi, L., & Abran, A. (2013). Promise and ISBSG Software Engineering Data Repositories: A Survey. *Joint Conference of the International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA)*. Ankara, Turkey.

Chen, H.-M., Kazman, R., Garbajosa, J., & Gonzalez, E. (2016). Toward big data value engineering for innovation. *ACM Proceedings of the 2nd International Workshop on BIG Data Software Engineering*.

Chidambara, R., & Senthil Kumar, M. (2016). Contribution of risk assessment techniques in effort estimation of software development process. *PARIPEX-Indian Journal of Research, 5*(4).

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *The Journal of Systems and Software, 81*, 961-971.

CMMI Product Team. (2010). *CMMI for Development, Version 1.3*. Hanscom: Carnegie Mellon.

COSMIC. (2011). *COSMIC-FFP: ISO/IEC 19761:2011 - Software engineering. A functional size measurement method*. London: Common Software Measurement International Consortium (COSMIC).

Czarnacka-Chrobot, B. (2009). The role of benchmarking data in the software development and enhancement projects effort planning. *Proceedings of the 2009 conference on New Trends in Software Methodologies, Tools and Techniques*.

Czarnacka-Chrobot, B. (2010). Rational pricing of business software systems on the basis of functional size measurement: a case study from Poland. *Proceedings 7th Software Measurement European Forum (SMEF)*. Rome, Italy.

Dagnino, A. (2013). Estimating Software-Intensive Projects in the Absence of Historical Data. *IEEE - International Conference on Software Engineering (ICSE)*. San Francisco, CA, USA.

Dekkers, C., & Forselius, P. (2010). Scope Management: 12 Steps for ICT Program Recovery. *CROSSTALK The Journal of Defense Software Engineering, January / February*, 16-21.

DeMarco, T. (1984). An algorithm for sizing software products. *ACM SIGMETRICS Performance Evaluation Review, 12*(2), 13-22.

Deng, K., & MacDonell, S. (2008). Maximising data retention from the ISBSG repository. *Proceedings of the twelfth International Conference on Evaluation and Assesment of Software Engineering (EASE)*.

Déry, D., & Abran, A. (2005). Investigation of the effort data consistency in the ISBSG repository. *Proceedings of the 15th Intern. Workshop on Software Measurement*.

Deursen, A. van, Klint, P., & Verhoef, C. (1999). *Research issues in the renovation of legacy systems*. Springer Berlin Heidelberg.

Devanbu, P., Zimmermann, T., & Bird, C. (2016). Belief & evidence in empirical software engineering. *Proceedings of the 38th international conference on software engineering*. ACM.

Díaz-Pace, J., Villavicencio , C., Schiaffino, S., & Nicoletti, M. (2016). Producing Just Enough Documentation: An Optimization Approach Applied to the Software Architecture Domain. *Journal on Data Semantics, 5*(1), 37-53.

Dingsøyr, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology, 77*, 56-60.

Dybå, T. (2003). Factors of Software Process Improvement Success in Small and Large Organizations: an Emperical Study in the Scandinavian Context. *ESEC/FSE*. Helsinki, Finland.

Dybå, T. (2005). An Empirical Investigation of the Key Factors for Success in Software Process Improvement. *IEEE Transactions on Software Engineering, 31*(5), 410-424.

Dybå, T., Kitchenham, B. A., & Jorgensen, M. (2005). Evidence-based software engineering for practitioners. *Software, IEEE, 22*(1), 58-65.

Dye, L., & Pennypacker, J. (1999). *Project Portfolio Management: Selecting and Prioritizing Projects for Competitive Advantage*. West Chester, Pennsylvania: Center for Business Practices.

El Emam, K., & Günes Koru, A. (2008). A replicated survey of IT software project failures. *IEEE software, 25*(5), 84-90.

Estler, H., Nordio, M., Furia, C. A., Meyer, B., & Scheider, J. (2012). Agile vs. Structured Distributed Software Development: A Case Study. *IEEE Seventh International Conference on Global Software Engineering (ICGSE)*. Porto Allegre, Brasil.

Eveleens, L., & Verhoef, C. (2009). Quantifying IT forecast quality. *Science of computer programming, 74*(11), 934-988.

Eveleens , L., & Verhoef, C. (2010). The rise and fall of the chaos report figures. *IEEE software, 27*(1), 30-36.

Faulk, S., Harmon, D., & Raffo, D. (2000). Value-Based Software Engineering (VBSE): A Value-Driven Approach to Product-Line Engineering. *First International Conference on Software Product-Line Engineering*.

Fernández-Diego, M., & González-Ladrón-de-Guevara, F. (2014). Potential and limitations of the ISBSG dataset in enhancing software engineering research: A mapping review. *Information and Software Technology, 56*(6), 527-544.

Ferreira, C., & Cohen, J. (2008). Agile systems development and stakeholder satisfaction: a South African empirical study. *ACM Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries*.

Feyh, M., & Petersen, K. (2013). Lean software development measures and indicators-a systematic mapping study. *Lean Enterprise Software and Systems*, 32-47.

Fink, L., & Lichtenstein, Y. (2014). Why project size matters for contract choice in software development outsourcing. *ACM SIGMIS Database, 45*(3), 54-71.

Fischman, L., McRitchie, K., & Galorath, D. (2005). Inside SEER-SEM. *Crosstalk - The Journal of Defense Software Engineering, April*, 26-28.

FiSMA. (2010). *FiSMA FSM: ISO/IEC 29881 - Information technology – Software and systems engineering – FiSMA 1.1 functional size measurement method*. Helsinki: Finnish Software Metrics User Association (FiSMA).

Fitzgerald, B., Stol, K., O'Sullivan, R., & O'Brien, D. (2013). Scaling agile methods to regulated environments: An industry case study. *35th International Conference on Software Engineering (ICSE)*. IEEE.

Fitzgerald, B., & Stol, K., (2015). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software, July*(Elsevier), 1-14.

Foss, T., Stensrud, E., Kitchenham, B., & Myrtveit, I. (2003). A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Transactions on Software Engineering, 29*(11), 985-995.

Gangadharan, G., Kuiper, E., Janssen, M., & Luttighuis, P. (2013). IT Innovation Squeeze: Propositions and a Methodology for Deciding to Continue or Decommission Legacy

Systems, Grand Successes and Failures in IT. *Public and Private Sectors, International Federation of Information Processing*, 481–494 .

Garre, M., Cuadrado, J., Sicilia, M., Charro, M., & Rodríguez, D. (2005). Segmented parametric software estimation models: Using the em algorithm with the isbsg 8 database. *Information Technology Interfaces*.

Gencel, C., & Demirors, O. (2008). Functional Size Measurement Revisited. *ACM Transactions on Software Engineering and Methodology, 17*(3), 15:1-15:36.

Gilb, T., & Finzi, S. (1988). *Principles of software engineering management* (Vol. 11). Reading, MA: Addison-Wesley.

Glass. (2002). *Facts and Fallacies of Software Engineering*. Addison Wesley.

Glass. (2005). IT Failure Rates-70% or 10-15%?. *IEEE Software, 22*(3), 110-112.

Glass. (2006). The Standish Report: Does It Really Describe a Software Crisis? *Communications of the ACM, 49*(8), 15-16.

Gong, Y., & Janssen, M. (2012). From policy implementation to business process management: Principles for creating flexibility and agility. *Government Information Quarterly, 29*, 61-71.

Green, P. (2011). Measuring the impact of scrum on product development at adobe systems. *IEEE 44th Hawaii International Conference on System Sciences (HICSS).*

Hall, T., Rainer, A., & Baddoo, N. (2002). Implementing Software Process Improvement: An Empirical Study. *Software Process Improvement and Practice, 7*, 3-15.

Hayes, B. (1998). *Measuring customer satisfaction: Survey design, use, and statistical analysis methods*. ASQ Quality Press.

Heemstra, F., & Kusters, R. (1989). Controlling Software Development Costs: A Field Study. *International Conference on Organisation and Information Systems*. Bled, Yugoslavia.

Heemstra, F., & Kusters, R. (1991). Function point analysis: Evaluation of a software cost estimation model. *European Journal of Information Systems, 1*(4), 223-237.

Heemstra, F. (1992). Software cost estimation. *Information and Software Technology, 34*(10), 627 - 639.

Hofner, G., Mani, V., Nambiar, R., & Apte, M. (2011). Fostering a high-performance culture in offshore software engineering teams using balanced scorecards and project scorecards. *IEEE Sixth International Conference on Global Software Engineering*.

Hopkins, W. (2000). A new view of statistics. *Internet Society for Sport Science*.

Huijgens, H., & van Solingen, R. (2013a). Measuring Best-in-Class Software Releases. *IWSM-MENSURA 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement*(IEEE), 137-146.

Huijgens, H., van Solingen, R., & van Deursen, A., (2013b). *How To Build a Good Practice Software Project Portfolio - Technical Report TUD-SERG-2013-019*. Delft, The Netherlands: Delft University of Technology.

Huijgens, H., & van Solingen, R. (2014a). A replicated study on correlating agile team velocity measured in function and story points. *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics (WETSoM).*

Huijgens, H., Gousios, G., & van Deursen, A. (2014b). *Pricing via Functional Size: A Case Study of 77 Outsourced Projects - Technical Report TUD-SERG-2014-012*. Delft, The Netherlands: Delft University of Technology.

Huijgens, H., van Solingen, R., & van Deursen, A. (2014c). How To Build a Good Practice Software Project Portfolio? *ICSE Companion 2014 Companion Proceedings of the 36th International Conference on Software Engineering (SEIP),* IEEE, 64-73.

Huijgens, H. (2015a). Evidence-Based Software Portfolio Management. *Doctoral Symposium of the 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).*

Huijgens, H., Bruntink, M., van Deursen, A., van der Storm, T., & Vogelezang, F. (2015b). An Exploratory Study on Automated Derivation of Functional Size based on Code. *ACM Proceedings of the International Conference on Software and Systems Process (ICSSP)* (pp. 56-65). Austin, Texas, USA: Delft University of Technology.

Huijgens, H., Gousios, G., & van Deursen, A. (2015c). Pricing via Functional Size - A Case Study of a Company's Portfolio of 77 Outsourced Projects. *ACM/IEEE 9th International Symposium on Empirical Software Engineering and Measurement (ESEM).* Beijing, China.

Huijgens, H. (2016a). Evidence-Based Software Portfolio Management: A Tool Description and Evaluation. *ACM Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. Limerick, Ireland.

Huijgens, H., & Vogelezang, F. (2016b). Do Estimators Learn? On the Effect of a Positively Skewed Distribution of Effort Data on Software Project Productivity. *ACM Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics (WETSoM)* (pp. 8-14). Austin, Texas, USA: Delft University of Technology.

Huijgens, H., van Deursen, A., & van Solingen, R. (2016c). An Exploratory Study on the Effects of Perceived Value and Stakeholder Satisfaction on Software Projects. *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE).* Limerick, Ireland.

Huijgens, H., van Deursen, A., Minku, L., & Lokan, C. (2016c). *Effort and Cost of Software Engineering: A Comparison of Two Industrial Data Sets - Technical Report TUD-SERG-2016-017.* Software Engineering Research Group (SERG), Delft University of Technology.

Huijgens, H., van Deursen, A., & van Solingen, R. (2016d). Success Factors in Managing Legacy System Evolution: A Case Study. *IEEE/ACM Proceedings of the International Conference on Software and System Processes (ICSSP).* Austin, TX, USA.

Huijgens, H. (2017a). EBSPM Research Repository. *4TU.ResearchData*. Delft University of Technology; https://data.4tu.nl/; DOI 10.4121/uuid:42fd1be1-325f-47a4-ba39-31af35ca7f75.

Huijgens, H., Lamping, R., Stevens, D., Rothengatter, H., Romano, D., & Gousios, G. (2017b). Strong Agile Metrics: Mining Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams. *ACM Prcodeedings of the 11th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE).* Paderborn, Germany.

Huijgens, H., van Deursen, A., Minku, L., & Lokan, C. (2017c). Effort and Cost of Software Engineering: A Comparison of Two Industrial Data Sets. *ACM 21st International Conference on Evaluation and assessment (EASE).* Karlskrona, Sweden.

Huijgens, H., van Deursen, A., & van Solingen, R. (2017d). The Effects of Perceived Value and Stakeholder Satisfaction on Software Project Impact. *Information and Software Technology*.

Huijgens, H., van Deursen, A., & van Solingen, R. (2017e). *TUD-SERG-2017-001 - The Effects of Perceived Value and Stakeholder Satisfaction on Software Project Impact - Technical Report*. Delft University of Technology.

Huisman, M., Bos, H., Brinkkemper, S., van Deursen, A., Groote, J., Lago, P., & Visser, E. (2016). Software that meets its Intent. *International Symposium on Leveraging Applications of Formal Methods*. Springer International Publishing.

IFPUG. (2009). *IFPUG FSM Method: ISO/IEC 20926 - Software and systems engineering – Software measurement – IFPUG functional size measurement method.* New York: International Function Point User Group (IFPUG).

International Standish Group. (1994). *The Chaos Report.*

ISBSG. (2014). *International Software Benchmarking Standards Group*. Retrieved from http://www.isbsg.org/

ISBSG, Jones, C., & Reifer Consultants. (n.d.). *The Impact of Software Size on Productivity.* ISBSG.

Janssen, M., & Klievink, B. (2010). ICT-project failure in public administration: The need to include risk management in enterprise architectures. *Proceedings of the 11th Annual International Conference on Digital Government Research on Public Administration Online: Challenges and Opportunities*. Digital Government Society of North America.

Janssen, M., & Estevez, E. (2013). Lean government and platform-based governance - Doing more with less. *Government Information Quarterly, 30*, 1-8.

Jeffery, M., & Leliveld, I. (2004). Best practices in IT portfolio management. *Sloan Management Review, 45*(3).

Jeffery, R., Ruhe, M., & Wieczore, I. (2000). A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data. *Information and software technology, 42*(14), 1009-1016.

Jeffery, R., Ruhe, M., & Wieczore, I. (2001). Using public domain metrics to estimate software development effort. *IEEE Seventh International Software Metrics Symposium (METRICS).*

Jiang, J., & Klein, G. (2000). Software development risks to project effectiveness. *Journal of Systems and Software, 52*(1), 3-10.

Jones, C. (1995). Patterns of large software systems: failure and success. *Computer*, 86-87.

Jones, C. (2000). *Software, Assessments, Benchmarks, and Best Practices.* New York: Addison Wesley Longman.

Jones, C. (2011). *Sources of Software Benchmarks.* Capers Jones & Associates.

Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *The Journal of Systems and Software, 70*(IEEE), 37-60.

Jørgensen, M., & Moløkken-Østvold, K. (2006). How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology, 48*(4), 297-301.

Jørgensen, M., & Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering, 33*(1), 33-53.

Jørgensen, M., & Gruschke, T. M. (2009). The impact of lessons-learned sessions on effort estimation and uncertainty assessments. *IEEE Transactions on Software Engineering, 35*(3), 368-383.

Jørgensen, M., Halkjelsvik, T., & Kitchenham, B. (2012). How does project size affect cost estimation error? Statistical artifacts and methodological challenges. *International Journal of Project Management, 30*, 839-849.

Jørgensen, M., & Kitchenham, B. (2012). Interpretation problems related to the use of regression models to decide on economy of scale in software development. *Journal of Systems and Software, 85*(11), 2494-2503.

Jørgensen, M. (2016). A survey on the characteristics of projects with success in delivering client benefits. *Information and Software Technology, 78*, 83-94.

Kan, S. (1995). *Metrics and Models in Software Quality Engineering*. Reading, Massachusetts: Addison Wesley Longman.

Kaplan, R., & Norton, D. (1995). *Putting the balanced scorecard to work.*

Kemerer, C. (1987). An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM, 30*(5), 416-429.

Kitchenham, B. A., Dybå, T., & Jørgensen, M. (2004). Evidence-based software engineering. *Proceedings of the 26th international conference on software engineering, IEEE Computer Society.*

Kitchenham, B., Pretorius, R., Budgen, D., Brereton, P., Turner, M., Niazi, M., & Linkman, S. (2010). Systematic literature reviews in software engineering–a tertiary study. *Information and Software Technology, 52*(8), 792-805.

Kocaguneli, E., Menzies, T., & Mendes, E. (2015). Transfer learning in effort estimation. *Empirical Software Engineering, 20*(3), 813-843.

Kupiainen, E., Mäntylä, M., & Itkonen, J. (2015). Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies. *Information and Software Technology, 62*, 143-163.

Lederer, A., & Prasad, J. (1993). Information systems software cost estimating: a current assessment. *Journal of Information Technology, 8*(Palgrave Macmillan), 22-33.

Lindberg, K. R. (1999). Software developer perceptions about software project failure: a case study. *Elsevier - The Journal of Systems and Software, 49*, 177-192.

Lo, D., Nagappan, N., & Zimmermann, T. (2015). How practitioners perceive the relevance of software engineering research. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM.

Lokan, C., Wright, T., Hill, P., & Stringer, M. (2001). Organizational benchmarking using the ISBSG data repository. *IEEE Software, 18*(5), 26-32.

Lokan, C., & Mendes, E. (2006). Cross-company and single-company effort models using the ISBSG database: a further replicated study. *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*.

Maciel, R., & Barros, M. (2016). Risk-Aware Multi-stakeholder Next Release Planning Using Multi-objective Optimization. *22nd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ).*

Madachy, R., Rosa, W., Boehm, B., Clark, B., & Tan, T. (2011). Us dod application domain empirical software cost analysis. *IEEE International Symposium onEmpirical Software Engineering and Measurement (ESEM).*

McDonald, J. (n.d.). *Handbook of Biological Statistics.* Retrieved 11 2016, from http://www.biostathandbook.com/multiplecomparisons.html

McFarlan, F. (1981, September-October). Portfolio Approach to Information Systems. *Harvard Business Review, 59*, 142-150.

Melo, C., Cruzes, D., Kon, F., & Conradi, R. (2011). Agile team perceptions of productivity factors. *IEEE Agile Conference (AGILE).*

Mendes, E., Lokan, C., Harrison, R., & Triggs, C. (2005). A replicated comparison of cross-company and within-company effort estimation models using the ISBSG database. *IEEE 11th IEEE International Symposium on Software Metrics.*

Menzies, T., & Shepperd, M. (2012). Special issue on repeatable results in software engineering prediction. *Empirical Software Engineering, 17*(1), 1-17.

Menzies, T., & Zimmermann, T. (2013). Software Analytics: So What? *IEEE Software, July / August*, 31-37.

Meyer, B. (2014). *Agile!: The Good, the Hype and the Ugly.* Springer Science & Business Media.

Minku, L., & Yao, X. (2013). Ensembles and Locality: Insight on Improving Software Effort Estimation. *Information and Software Technology, Special Issue on Best Papers from PROMISE 2011, 55*(5), 1512-1528.

Minku, L., Mendes, E., & Ferrucci, F. (2015). How to make best Use of Cross-Company Data for Web Effort Estimation? *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* .

Minku. (2016). On the Terms Within- and Cross-Company in Software Effort Estimation. *ACM Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering.*

Misra, S., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *The Journal of Systems and Software, 82*, 1869-1890.

Moløkken, K., & Jørgensen, M. (2003). A Review of Surveys on Software Effort Estimation. *IEEE - Proceedings of ISESE - International Symposium on Empirical Software Engineering*, 223-230.

Murphy-Hill, E., Parnin, C., & Black, A.P. (2012). How We Refactor, and How We Know It. *IEEE Transactions on Software Engineering*, 5-18, vol. 38, No. 1.

Murphy-Hill, E., Zimmermann, T. & Nagappan, N. (2014). Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? *ACM - Proceedings of the 36th International Conference on Software Engineering (ICSE)*, 1-11, Hyderabad, India.

Nakai, H., Tsuda, N., Honda, K., Washizaki, H., & Fukazawa, Y. (2016). Initial framework for software quality evaluation based on iso/iec 25022 and iso/iec 25023. *International Conference on Software Quality, Reliability and Security Companion (QRS-C).* IEEE.

Nazir, N., Hasteer, N., & Bansal, A. (2016). A survey on agile practices in the Indian IT industry. *IEEE 6th International Conference-Cloud System and Big Data Engineering (Confluence).*

NESMA. (2004). *NESMA functional size measurement method conform ISO/IEC 24570, version 2.2*. Netherlands Software Measurement User Association (NESMA).

NESMA. (2005). *Nesma functional size measurement method conform ISO/IEC 24570, version 2.1*. Netherlands Software Measurement User Association (NESMA).

Niazi, M., Wilson, D., & Zowghi, D. (2003). A maturity model for the implementation of software process improvement: an empirical study. *The Journal of Systems and Software*.

Niazi, M., Wilson, D., & Zowghi, D. (2006). Critical Success Factors for Software Process Improvement Implementation: An Empirical Study. *Software Process Improvement and Practice, 11*, 193-211.

Nisbet, R., Miner, G., & Elder IV, J. (2009). *Handbook of Statistical Analysis and Data Mining Applications*. Academic Press.

Nokes, S. (2007). *The Definitive Guide to Project Management* (Vol. 2nd Edition). London: Financial Times / Prentice Hall. doi:ISBN 978-0-273-71097-4

Object Management Group (OMG). (2014). *Automated Function Points (AFP)*. Formal/2014-01-03 - Version 1.0.

Oligny, S., Bourque, P., Abran, A., & Fournier, B. (2000). Exploring the relation between effort and duration in software engineering projects. *Proceedings of the World Computer Congress*, (pp. 175-178).

Pascoe, G. (1983). Patient satisfaction in primary health care: A literature review and analysis. *Eval Prog Planning, 6*, 185-201.

Passos, C., Braun, A., Cruzes, D., & Mendonca, M. (2011). Analyzing the impact of beliefs in software project practices. *IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*.

Pekki, J. (2016). How the Company Manages Critical Success Factors in Software Process Improvement Initiatives: Pilot Case-Study in Finnish Software Company." European Conference on Software Process Improvement. Springer International Publishing, 2016. *European Conference on Software Process Improvement*. Springer International Publishing.

Pendharkar, P., & Rodger, J. (2009). The relationship between software development team size and software development cost. *Communications of the ACM, 52*(1), 141-144.

Petersen, K. (2011). Measuring and predicting software productivity: A systematic map and review. *Information and Software Technology, 53*(4), 317-343.

Pitangueira, A., Tonella, P., Susi, A., Maciel, R., & Barros, M. (2016). Risk-Aware Multi-stakeholder Next Release Planning Using Multi-objective Optimization. *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer International Publishing.

Premrai, R., Shepperd, M., Kitchenham, B., & Forselius, P. (2005). An Empirical Analysis of Software Productivity Over Time. *IEEE International Symposium Software Metrics*. Como, Italy.

Procaccino, J., Verner, J., & Overmyer, S. (2002). Case Study: Factors for Early Prediction of Software Success & Failure. *Elsevier - Information and Software Technology, 44*(1), 53-62.

Putnam, L., & Meyers, W. (2003). *Five Core Metrics, The Intelligence Behind Succesfull Software Management*. New York: Dorset House Publishing.

Radliński, Ł. (2011). Factors of Software Quality–Analysis of Extended ISBSG Dataset. *Foundations of Computing and Decision Studies, 36*(3-4), 293-313.

Raemaekers, S., van Deursen, A., & Visser, J. (2015). *Origin, Impact and Cost of Interface Instability (doctoral thesis).* doi:doi:10.4233/uuid:dbb70852-e06b-40f7-b872-60047f962dbc

Rainer, A., & Hall, T. (2002). Key success factors for implementing software process improvement: a maturity-based analysis. *Journal of Systems and Software, 62*(2), 71-84.

Ramasubbu, N., & Balan, R. (2012). Overcoming the challenges in cost estimation for distributed software projects. *Proceedings of the 34th International Conference on Software Engineering. IEEE.*

Ramasubbu, N., Cataldo, M., Balan, R., & Herbsleb, J. (2011). Configuring global software teams: a multi-company analysis of project productivity, quality, and profits. *Proceedings of the 33rd international conference on Software engineering. ACM.*

Reel, J. (1999). Critical Success Factors in Software Projects. *IEEE Software, May-June*, 18-23.

Reyck, B. d., Grushka-Cockayne, Y., Lockett, M., Calderini, S., Moura, M., & Sloper, A. (2005). The impact of project portfolio management on information technology projects. *International Journal of Project Management, 23*(7), 524-537.

Rubinstein, D. (2007). Standish group report: There's less development chaos today. *Software Development Times, 1.*

Rumsey, D.J. (2016). How to Interpret a Correlation Coefficient r. *Statistics For Dummies, 2nd Edition*. http://www.dummies.com/education/math/statistics/how-to-interpret-a-correlation-coefficient-r/

Runeson, P., Host, M., Rainer, A., & Regnell, B. (2012). *Case Study Research in Software Engineering; Guidelines and Examples*. Hoboken, New Jersey. USA: John Wily & Sons.

Sauer, C., & Cuthbertson, C. (2003). The State of IT Project Management in the UK 2002–2003. *Computer Weekly, April.*

Schmidt, R., Lyytinen, K., Cule , P., & Keil, M. (2001). Identifying software project risks: An international Delphi study. *Journal of management information systems, 17*(4), 5-36.

Schwaber, K., & Sutherland, J. (2011). *The Scrum Guide*. (Scrum Alliance 21) Retrieved from www.scrumguides.org

Shah , S., Papatheocharous, E., & Nyfjord, J. (2015). Measuring productivity in agile software development process: a scoping study. *ACM Proceedings of the 2015 International Conference on Software and System Process (ICSSP).*

Shepperd, M. (2014). Cost prediction and software project management. In *Software Project Management in a Changing World* (pp. 51–71). Springer.

Singh, B., Punhani, A., & Misra, A. (2016). Integrated Approach of Software Project Size Estimation. *International Journal of Software Engineering and Its Applications, 10*(2), 45-64.

Šmite, D., Calefato, F., & Wohlin, C. (2015). Cost-Savings in Global Software Engineering: Where's the Evidence. *IEEE Software, 32*(4), 26-32.

Šmite, D., & van Solingen, R. (2016, Sept-Oct). What's the True Hourly Cost of Offshoring? *IEEE Software, 33*(5), 60-70.

Solingen, R. van (2004). Measuring the ROI of software process improvement. *IEEE Software, 21*(3), 32-38.

Solingen, R. van & Berghout, E. (1999). *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill.

Sonnekus, R., & Labuschagne, L. (2004). Establishing the Relationship between IT Project Management Maturity and IT Project Success in a South African Context," Proc. 2004. *PMSA Global Knowledge Conf., Project Management South Africa*, 183-192.

Stelzer, D., & Mellis, W. (1998). Success Factors of Organizational Change in Software Process Improvement. *Software Process Improvement and Practice, 4*, 227-250.

Strand, K., & Karlsen, K. (2014). Agile Contracting and Execution. *PROMIS*.

Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. (2007). Distributed Scrum: Agile Project Management with Outsourced Development Teams. *40th International Conference on System Sciences*. Hawaii.

Tihinen, M., Parviainen, P., Suomalainen, T., & Karhu, K. (2011). ABB Experiences of Boosting Controlling and Monitoring Activities in Collaborative Production. *6th IEEE International Conference on Global Software Engineering (ICGSE)*. Helsinki.

Turhan, B., & Mendes, E. (2014). A comparison of cross- versus single- company effort prediction models for web projects. *Euromicro Conference on Software Engineering and Advanced Applications*. Verona, Italy.

UKSMA. (2002). *Mk II Function Point Analysis: ISO/IEC 20968 - Software engineering – Ml II Function Point Analysis – Counting Practices Manual*. London: UK Software Metrics Association (UKSMA).

Valerdi, R. (2011). Convergence of expert opinion via the wideband delphi method: An application in cost estimation models. *Incose International Symposium*. Denver, USA.

Verhoef, C. (2002). Quantitative IT Portfolio Management. *Elsevier - Science of Computer Programming, 45*(1), 1-96.

Wieczorek, I., & Ruhe, M. (2002). How valuable is company-specific data compared to multi-company data for software cost estimation? *IEEE Proceedings of the Eighth IEEE Symposium on Software Metrics*.

Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., Regnell, B., & Wesslen, A. (2000). *Experimentation in Software Engineering*. Heidelberg: Springer.

Wu, C., Naqibuddin, M., & Fleisher, L. (2001). Measurement of patient satisfaction as an outcome of regional anesthesia and analgesia: A systematic review. *Reg Anesth Pain Med, 26*, 196-208.

Yin, R. (2008). *Case Study Research - Design and Methods*. Los Angelos, USA: Sage Publications.

Zowghi, D., & Nurmuliani., N. (2002). A study of the impact of requirements volatility on software project performance. *IEEE Ninth Asia-Pacific Software Engineering Conference,*.