Department of Computer Science and Engineering

Development and evaluation of a lightweight root cause analysis method in software project retrospectives

Timo O.A. Lehtinen

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 159/2014

Development and evaluation of a lightweight root cause analysis method in software project retrospectives

Timo O.A. Lehtinen

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at Auditorium T2 of the school on 19 December 2014 at 12 noon.

Aalto University
School of Science
Department of Computer Science and Engineering
Software Process Research Group

Supervising professor
Professor Casper Lassenius

Thesis advisors
Professor Mika V. Mäntylä
Postdoctoral researcher Juha Itkonen

Preliminary examiners
Professor Torgeir Dingsøyr
Norwegian University of Science and Technology, Norway

Professor Jürgen Münch
University of Helsinki, Finland

Opponents
Professor Magne Jørgensen
University of Oslo, Norway

Aalto University publication series
DOCTORAL DISSERTATIONS 159/2014

ISBN 978-952-60-5907-5 (printed)
ISBN 978-952-60-5908-2 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-5908-2

Unigrafia Oy
Helsinki 2014

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Timo O.A. Lehtinen

Name of the doctoral dissertation
Development and evaluation of a lightweight root cause analysis method in software project retrospectives

Publisher School of Science

Unit Department of Computer Science and Engineering

Series Aalto University publication series DOCTORAL DISSERTATIONS 159/2014

Field of research Software Engineering

Manuscript submitted 6 June 2014 **Date of the defence** 19 December 2014

Permission to publish granted (date) 9 September 2014 **Language** English

☐ **Monograph** ☒ **Article dissertation (summary + original articles)**

Abstract

Software projects are famous for their problems. The most common problems include low software quality and schedule and cost overruns. According to the theory of causality, problems escalate from the causal relationships of mutually exclusive events. The consequences of the problems motivate software companies to improve their work practices.

Improvement of work practices is based on the detection of occurred problems and the analyses of their underlying causes. "Retrospective" is a term that refers to post-project activities where the occurred problems are considered in order to make improvements. Software project retrospectives include the detection of the occurred problems, reasoning their causes, and developing corrective actions. There are many retrospective methods that follow these three phases.

Root cause analysis (RCA) is a structured investigation of problems to detect their underlying causes. This dissertation considers the applicability of RCA in the retrospectives of small- and medium-sized software product organizations. The research focuses on three research problems. The first problem is to explain how to conduct RCA in collocated and distributed software project retrospectives. The second problem is to study whether RCA is perceived as efficient and easy to use. The third problem is to consider whether the outcome of RCA indicates how the causes of project failures are interconnected.

In this dissertation, an RCA method (ARCA) is developed. Thereafter, it is evaluated in a total of six industrial cases and in one controlled student experiment. Additionally, a software tool, called ARCA-tool, for improving the ARCA method is developed. The tool is evaluated in two industrial cases. The ARCA method includes four steps: target problem detection, root cause detection, corrective action innovation, and documentation of results. It requires approximately five hours of team work. A total of 97 participants evaluated the ARCA method and it was used in a total of 19 software project retrospectives.

One of the key contributions of this dissertation is the ARCA method, which is applicable to collocated and distributed retrospectives with the assistance of ARCA-tool. Another key contribution is the empirical evaluation of the ARCA method. The method is perceived as useful, easy to use, and cost-efficient for detecting the causes of software project problems and developing corrective actions. Furthermore, the method helps to understand what happened, where it happened, and why it happened. It reveals interconnections between software process areas, which is important for process-improvement activities.

Keywords root cause analysis, software engineering, retrospective, software project failure, usefulness, cost-efficiency, ease of use, field study, case study, controlled experiment, design science

ISBN (printed) 978-952-60-5907-5 **ISBN (pdf)** 978-952-60-5908-2

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2014

Pages 202 **urn** http://urn.fi/URN:ISBN:978-952-60-5908-2

Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Timo O.A. Lehtinen

Väitöskirjan nimi
Kevyen juurisyyanalyysimenetelmän kehitys ja evaluointi ohjelmistoprojektien retrospektiiveissä

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietotekniikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 159/2014

Tutkimusala Ohjelmistotuotanto

Käsikirjoituksen pvm 06.06.2014 **Väitöspäivä** 19.12.2014

Julkaisuluvan myöntämispäivä 09.09.2014 **Kieli** Englanti

☐ **Monografia** ☒ **Yhdistelmäväitöskirja (yhteenveto-osa + erillisartikkelit)**

Tiivistelmä

Ohjelmistoprojektit ovat kuuluisia ongelmistaan. Yleisimpiä ongelmia ovat tuotoksien heikko laatu, pitkittyneet toimitusajat, sekä ylittyneet kustannukset. Kausaalisuusteorian mukaan ongelmat muodostuvat toisensa poissulkevien tapahtumien välisistä kausaalisista suhteista. Ongelmien seuraukset motivoivat ohjelmistoyrityksiä kehittämään toimintatapojaan.

Toimintatapojen kehittäminen perustuu tapahtuneiden ongelmien tunnistamiseen sekä niiden perimmäisten syiden tulkintaan. "Retrospektiivi" on termi, jolla viitataan projektin jälkeiseen aktiviteettiin, joka tähtää toiminnan kehittämiseen reflektoimalla toiminnassa ilmenneitä ongelmia. Ohjelmistoprojektien retrospektiivit sisältävät kolme vaihetta: ongelmien tunnistaminen ja niiden syiden tulkinta, sekä ratkaisuideoiden kehittäminen. Useat retrospektiivimenetelmät seuraavat näitä vaiheita.

Juurisyyanalyysi (RCA) on rakenteellinen tapa tunnistaa ongelmien taustalla piileviä syitä. Tässä väitöskirjassa tutkitaan RCA:n soveltuvuutta pienten ja keskisuurten ohjelmistotuoteorganisaatioiden retrospektiiveihin. Työ tarkastelee kolmea tutkimusongelmaa. Ensimmäisenä pyritään ymmärtämään miten RCA:ta voidaan käyttää kasvotusten tapahtuvissa että fyysisesti hajautetuissa retrospektiiveissä. Toisena ongelmana on tutkia koetaanko RCA tehokkaaksi ja helppokäyttöiseksi lähestymistavaksi. Kolmas ongelma pohtii indikoiko RCA:n tuottama informaatio kuinka epäonnistuneen projektin aiheuttajat liittyvät toisiinsa.

Työssä kehitetään RCA-menetelmä (ARCA), jota evaluoidaan kuuden teollisen tutkimuksen sekä yhden kontrolloidun opiskelijakokeen avulla. Työssä kehitetään myös ohjelmistotyökalu (ARCA – työkalu) tehostamaan ARCA – menetelmää, jota evaluoidaan kahdessa teollisessa tutkimuksessa. Menetelmä sisältää neljä vaihetta: kohdeongelman tunnistaminen, juurisyiden tunnistaminen, korjaavien toimenpiteiden kehittäminen, sekä tuloksien dokumentointi. Sen soveltaminen vaatii noin viisi tuntia ryhmätyötä. Yhteensä 97 osallistujaa arvioi menetelmää ja sitä sovellettiin 19 ohjelmistoprojektin retrospektiivissä.

Eräs työn tärkeimmistä kontribuutioista on ARCA-menetelmä, joka yhdessä ARCA – työkalun kanssa soveltuu sekä kasvotusten tapahtuvaan että hajautettuun retrospektiiviin. Toinen tärkeä kontribuutio on menetelmän empiirinen arviointi. ARCA koetaan hyödylliseksi, helppokäyttöiseksi ja kustannustehokkaaksi tavaksi tunnistaa projektien ongelmien syitä sekä kehittää niihin ratkaisuideoita. Se auttaa ymmärtämään mitä tapahtui, sekä missä ja miksi tapahtui. Se paljastaa prosessialueiden välisiä suhteita, joka on tärkeää prosessinkehitysaktiviteeteissa.

Avainsanat juurisyyanalyysi, ohjelmistotuotanto, retrospektiivi, ohjelmistoprojektin epäonnistuminen, hyödyllisyys, kustannustehokkuus, helppokäyttöisyys, kenttätutkimus, tapaustutkimus, kontrolloitukoe, suunnittelutiede

ISBN (painettu) 978-952-60-5907-5 **ISBN (pdf)** 978-952-60-5908-2

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Helsinki **Painopaikka** Helsinki **Vuosi** 2014

Sivumäärä 202 **urn** http://urn.fi/URN:ISBN:978-952-60-5908-2

To Maria

Acknowledgements

Dear Reader, thank you for your interest on this doctoral dissertation. I hope you will find the work interesting and useful for your needs. Before considering the ideas and results, however, I wish you could use a few moments for the acknowledgments presented below.

This work was done in the Aalto University department of Computer Science and Engineering. The daily work was conducted in the research projects of Software Process Research Group funded by TEKES and industry partners. Part of this work was also funded by SoSE (Doctoral Programme on Software and Systems Engineering). These organizations deserve my acknowledgements. They have enabled the doctoral dissertations of many others, too.

Dear professor Casper Lassenius, the head of Software Process Research Group, you provided me the necessary funding and supported me with interesting insights and critical comments, which greatly improved this dissertation. You also set me tough goals and schedules that forced me keep on going.

Dear professor Mika V. Mäntylä and postdoctoral researcher Juha Itkonen, I truly admire your in-depth knowledge about software engineering. Your creativity is beyond comparison. Mika, you helped me through the research articles. Juha, you provided valuable instructions and academic guidelines especially in the latter parts of the dissertation.

I also got help from the researches of Software Business and Engineering Institute. Dear SoberIT people, you reviewed my articles and supported me by being yourself. Thank you, Jari Vanhanen, Kristian Rautiainen, Jarno Vähäniitty, Ville Heikkilä, Maria Paasivaara, and many others. Either should be forgotten the co-authors who greatly improved the articles. Thank you, Mika V. Mäntylä, Jari Vanhanen, Casper Lassenius, Juha Itkonen, Risto Virtanen, and Juha Viljanen. It has been a pleasure to work with you.

Furthermore, our industrial partners made this work possible and reasonable. They opened the doors for real-world software engineering, which made my observations and research ideas possible. I also want to thank the anonymous software engineering students who participated in my research as subjects. Additionally, I want to thank the experienced software engineering students who participated in the development of ARCA-tool. Thank you Risto Virtanen, Juha Viljanen, Helin Anssi Matti, Hovi Roope, Jaanto Jari, Kekäle Mika, Kere Markus, Koistinen Joona, Laukkanen Eero, Patana Jussi, Rihtniemi Pekka, Saarinen Jerome, Sevenius Toni, Valjus Mikko, and Viitanen Jonne.

I also acknowledge all of my friends and family members supporting me, and my closest ones, during this work. Especially, thank you Lasse Makkonen, Jarkko & Anneli Lehtinen, and Pirkka T. Pekkarinen whose experience and personal example guided me to finalize this project.

Finally, the greatest support came from my daughter and wife. Dear Iia Lehtinen, your birth is the moment of my life and it encouraged me to finalize this work. Every single breath you have taken has also changed my life. You have truly told me what to do next. Dear Maria Lehtinen, this thesis would not have been made without your love, support, and understanding. Your encouraging words and the newer ending trust on my work were the success factors of this dissertation. This book is dedicated to you. I love you from the bottom of my heart.

Espoo, October 2014

Timo Lehtinen

Timo Olli Antero Lehtinen

List of publications

This doctoral dissertation consists of two parts, a summary, and of the following articles which are referred to in the text by their numerals (I-V).

I **Development and evaluation of a lightweight root cause analysis method (ARCA method) – Field studies at four software companies**
Timo O.A. Lehtinen, Mika V. Mäntylä and Jari Vanhanen
Journal of Information and Software Technology, Volume 53, Issue 10, October 2011, Pages 1045–1061.

II **What are problem causes of software projects? Data of root cause analysis at four software companies**
Timo O.A. Lehtinen and Mika V. Mäntylä
Proceedings of International Symposium on Empirical Software Engineering and Measurement, 2011, Pages 388–391.

III **A tool supporting root cause analysis for synchronous retrospectives in distributed software teams**
Timo O.A. Lehtinen, Risto Virtanen, Juha O. Viljanen, Mika V. Mäntylä and Casper Lassenius
Journal of Information and Software Technology, Volume 56, Issue 4, April 2014, Pages 408–437.

IV **Perceived causes of software project failures – An analysis of their relationships**
Timo O.A. Lehtinen, Mika V. Mäntylä, Jari Vanhanen, Juha Itkonen and Casper Lassenius
Journal of Information and Software Technology, Volume 56, Issue 6, June 2014, Pages 623–643.

V **An experimental comparison of using cause-effect diagrams and simple memos in software project retrospectives**
Timo O.A. Lehtinen, Mika V. Mäntylä, Juha Itkonen and Jari Vanhanen
Journal of Systems and Software (2014), 26 pages, in revision.

Author's contribution

In articles I–V, the author contributed significantly to the creation of the research ideas. He also contributed significantly to the data collection and analyses including revealing the main findings for conclusions. Additionally, he wrote the original manuscripts. The co-authors provided comments, improvement ideas and criticisms for each article. Additionally, they helped to refine the text by changing wording, providing clarifications, adding some references, improving argumentation, and refining the discussion.

Article I: *Development and evaluation of a lightweight root cause analysis method (ARCA method) – Field studies at four software companies*
The author developed the ARCA method and conducted the literature review. He also collected and analyzed the data from the industrial cases. The co-authors participated in the observations and they provided minor improvement ideas for the ARCA method.

Article II: *What are problem causes of software projects? Data of root cause analysis at four software companies*
The author collected and analysed the research data. The co-author participated in the interpretation of results.

Article III: *A tool supporting root cause analysis for synchronous retrospectives in distributed software teams*
The author steered the development of the software tool and provided its requirements. He also observed the industrial data collection and wrote down notes for the data analysis. He conducted the data analyses.

Article IV: *Perceived causes of software project failures – An analysis of their relationships*
The author conducted the data collection and analysis. The results were interpreted together with the co-authors. Additionally, one co-author helped the author to conduct inter-rater agreement on the results.

Article V: *An experimental comparison of using cause-effect diagrams and simple memos in software project retrospectives*
The author conducted the data collection and analysis. The co-authors participated in the interpretation of results.

Terminology and abbreviations

ARCA	The RCA method that was developed in this study.
Bridge cause	A cause which is related to another process area than the one of its effect. A bridge cause explains how two process areas are related to one another.
Cause-effect diagram	A diagram of causes and effects including two types of causal structures. List-based structures: a fishbone diagram, a fault tree diagram, a logic tree, and a causal factor chart. Network-based structures: a directed graph and a matrix diagram.
Causal relationship	A cause-and-effect relationship between two mutually exclusive events, i.e. a cause and its effect.
Causal structure	An assembly of causes and effects which structures their mutual relationships.
Causal model	A complete specification of the causal relationships that govern a given domain.
Cause entity	An entity of causes and effects that are reasonable to process together.
Cause type	Expresses what the cause is. For example, a cause "There is lack of instructions on what I should do" has a type "People".
Cause sub-type	More detailed description of the cause type. For example, "instructions & experiences" is a sub-type of People.
Characteristic of detected cause (CDC)	A combination of the process area and cause type for an individual cause.
Depth level	The number of cause-effect pairs from a cause to the target problem. For example, Depth level=1 indicates causes which directly explain the target problem and Depth level=2 indicates the causes which explain the causes having Depth level=1.

Hub cause	A sub-cause which explains more than one cause. See also NoH.
Method effectiveness (ME)	ME indicates the number of detected causes per time unit.
Number of hub causes (NoH)	The number of hub causes in a causal structure.
Perception of participants (PP)	PP reflects the evaluations of participants.
Process area	An area of work and responsibility which represents one part of the whole software development process. For example, software testing.
Proposed cause	A cause of an event which is proposed for process improvement activities. See also Selected cause.
Size of depth levels, SoDL(x)	SoDL(x) is a function that indicates the number of causes in a depth level x. See also Depth level.
Software project failure	A recognizable failure to succeed in the cost, schedule, scope, or quality goals of the project. The "recognizable" means a failure perceived as severe enough to be prevented in the upcoming projects.
Software project retrospective	A post-project activity where a group of people "looks back" on the software project in order to facilitate learning and improvements based on the experiences gained during the project. Post-project review is a synonym for a retrospective.
Root cause analysis (RCA)	A structured investigation of a problem which takes a problem as an input and provides a set of its causes as an output.
RCA facilitator	A person who leads an RCA team.
Root cause	An underlying cause of the target problem that the management has the power to control.
Selected cause	A cause of an event which is selected for software process improvement activities. See also Proposed cause.
Sub-cause	A cause which explains another cause.
Target problem	A problem which is analysed by using RCA.

Table of contents

Part I: Summary

1. Introduction

"Everything that exists, and everything that happens, exists or happens as a necessary consequence of a previous state of things."

—T. N. Thiele (1931)

The discipline of today's software engineering (SE) originates from the software project problems introduced in the late 1960s (Naur and Randel 1969). Up to 34 percent of today's software projects are either unsuccessful or cancelled (El Emam and Koru 2008). Software project retrospectives have been used to increase the success rate of upcoming software projects. Software project retrospectives are post-project activities wherein a group of people "looks back" to the software project in order to facilitate learning and make improvements based on the experiences gained during the project (Birk, Dingsøyr, and Stålhane 2002).

1.1 Motivation

Root cause analysis (RCA) is a structured investigation of a problem to detect the underlying causes that need to be prevented (Latino and Latino 2006). It is a commonly recommended technique for problem prevention (Latino and Latino 2006; Andersen and Fagerhaug 2006; Ammerman 1998; Cooke 2003; Rooney and Vanden Heuvel 2004). It takes the problem as an input and provides a set of its perceived causes including the perceived causal relationships as an output. A causal relationship refers to the causal relationship between the cause and its effect (Chillarege et al. 1992).

In the SE context, RCA has been introduced as a method for software project retrospectives (Dingsøyr, Moe, and Nytrø 2001). It has been claimed to help in developing effective corrective actions (Rooney and Vanden Heuvel 2004). For example, a 50 % decrease in defect rates (Card 1998), a 53 % savings in costs, and a 24 % increase in productivity (Leszak, Perry, and Stoll 2000) have been reported. However, the work practices of the RCA methods have been introduced on too general a level to be adopted as such. Additionally, the subject matter experts' perceptions of the RCA methods have not been studied systematically. Furthermore, its added value for software project retrospectives has not been widely studied.

1.2 Study objectives

The main objective of this thesis is to develop and evaluate a lightweight RCA method and software tool for software project retrospectives in order to provide empirical evidence on its feasibility for software project failure prevention in small- and medium-sized (SME) organizations. Most of the prior studies on RCA have been conducted in large organizations (Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Gupta et al. 2008; Grady 1996; Mays 1990), but it could be useful in SME organizations too, as also noted by Stålhane et al. (2003). However, the optimal RCA method for SME organizations is likely different than the one for large organizations. Therefore, studying how to use RCA in the retrospectives of SME organizations is reasonable.

The thesis contributes to three research problems, which are introduced thoroughly in Section 2.4. The first research problem is to explain *how to conduct RCA in collocated and distributed software project retrospectives?* The second research problem is to study *whether RCA is perceived as efficient and easy to use in software project retrospectives?* The third research problem is to determine *whether the outcome of RCA indicates how the causes of software project failures are interconnected?*

While considering the research problems listed above, the thesis makes three scientific contributions. The first contribution is the lightweight RCA method and supporting software tool. These two artefacts contribute to the first research problem as they introduce how RCA can be conducted in collocated and distributed software project retrospectives (see articles I and III).

The second contribution is an empirical evaluation of the lightweight RCA method and software tool (see articles I, III, and V). The empirical evaluation contributes to the second research problem as it introduces how the software engineering practitioners perceived the ease of use, cost-efficiency, and outcome of the RCA method and its software tool. The empirical evaluation is divided into various levels of software project retrospectives. These include team-level retrospectives, organization-level retrospectives, and company-level retrospectives, ultimately aiming to reveal the causes of software project failures. Additionally, the evaluation covers the use of collocated and distributed retrospectives.

The third contribution is a detailed, in-depth analysis of the outcome of RCA (see articles II and IV). The analysis is limited to four cases of software project failures. The outcome analysis contributes to the third research problem as it provides empirical evidence on the feasibility of using RCA to explain why a software project failed, additionally, where the causes of the failure occurred and how the causes were related to one another.

The overall research approach in this thesis is design science (Hevner et al. 2004) including empirical evaluation with the mixed-methods approach (Shull, Sjøberg, and Singer 2008) that combines three main research approaches: observation-based industrial field studies (Lethbridge, Elliott Sim, and Singer 2005), case studies (Yin 1994), and controlled experiments (Juristo and Moreno 2003).

1.3 Structure of the thesis

This thesis consists of two parts: the dissertation summary and research articles. The dissertation summary starts with a brief introduction to the related work in Section 2, which includes the theorization and the use of RCA in SE context. The section ends with a discussion of the three research problems addressed in this thesis. Thereafter, Section 3 presents the research objectives and methods including the use of the design science framework. Section 4 introduces the lightweight RCA method, and Section 5 presents the developed software tool. The results of the empirical evaluation and the in-depth analysis of the RCA method outcome are summarized in Section 6. Section 7 discusses the research questions, implications, and threats to validity. Finally, Section 8 states the conclusions and directions for future work. The second part includes the research articles.

2. Related work

This section starts with the law of causality and discusses its relevance to analysing the causes of SE problems. Thereafter, the concept of RCA and an explanation of how it is used in software project retrospectives are introduced. The section ends with a discussion of gaps in the existing research.

2.1 The law of causality in software engineering

The underlying theory of problem prevention is based on the law of causality, which has been considered by scientists and philosophers starting with Aristotle (Álvarez 2009), Hume (1896), and recently Pearl (2000). The law of causality states that the occurrence of problems is the consequence of a previous state of actions (Thiele 1931). *Causality* refers to the causal relationship between sequential and mutually exclusive events (Granger 1988), i.e. the relationship between a cause and its effect (Chillarege et al. 1992). A c*ausal model* refers to "a complete specification of the causal relationships that govern a given domain" (Galles and Pearl 1997), i.e. it explains what happened, where it happened, and why it happened.

I make three assumptions based on the law of causality. First, the problems of software projects follow the law of causality. This assumption is logical because the software development work is based on sequential and mutually exclusive events, a set of linked activities (Wang and King 2000) in which the previous state of actions affect the latter state of actions. Thus, the law of causality exists also with software project problems. Prior studies support this assumption. Cerpa and Verner (2009) presented that causal relationships between the causes of software project failures likely exist. McLeod and MacDonell (2011) presented that the factors of the software project outcome are interconnected through multidimensional relationships. Furthermore, Xiangnan et al. (2010) presented that the causes of software project failures are caused by actions being interconnected through "internal" and "external" causes.

Second, the problems of software projects are interconnected over the process areas. This assumption divides the sequential and mutually exclusive events of software development into "software process areas", logical areas of different types of software development work. The prior studies indicate that software development process areas are interconnected (Monteiro et al. 2010). Therefore, it is reasonable to assume that a problem in one process area could also cause problems in other process areas.

Third, the problems of software projects reoccur in future projects if the related causal relationships are not detected and controlled. Prior studies have found many causes common in software project failures, which mean that the causes of failures transfer from prior projects to upcoming projects if they are not controlled or eliminated (Card 1998). Respectively, controlling the causes of problems has been introduced as valuable (Dingsøyr, Moe, and Nytrø 2001; Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Grady 1996; Kalinowski, Travassos, and Card 2008; Bjørnson, Wang, and Arisholm 2009; Al-Mamory and Zhang 2009; Siekkinen et al. 2008; Traeger, Deras, and Zadok 2008; Stålhane 2004; Bhandari et al. 1993; Jin et al. 2007). Thus, detecting and controlling the causal relationships of common software project problems is also practically useful.

2.2 Definitions of root cause analysis

In the terminology of this thesis, root cause analysis is a systematic process of detecting a target problem, detecting and organizing its causes, and recognizing its root causes. This definition considers the use of RCA as a technique for detecting the causes of a problem. In the prior literature, RCA has been introduced as a method for decreasing the likelihood of the reoccurrence of the problems (Rooney and Vanden Heuvel 2004; Card 1998; Leszak, Perry, and Stoll 2000; Card 1993). However, there seems to be a slight disagreement whether RCA is considered only a structured investigation of a problem (Latino and Latino 2006; Ammerman 1998; Leszak, Perry, and Stoll 2000; Bjørnson, Wang, and Arisholm 2009) or whether it also includes the development of corrective actions (Andersen and Fagerhaug 2006; Rooney and Vanden Heuvel 2004; Card 1998; Card 1993).

Furthermore, the prior literature has introduced a term, “root cause”, which has been used to indicate a target problem cause, which is perceived as important to control and feasible for process improvement activities. Conceptually, a target problem could be affected with numerous root causes. In the terminology of this thesis, a root cause is *an underlying cause of the target problem that the management has the power to control*. This definition considers the root cause as an internal cause of the company (Xiangnan, Hong, and Weijie 2010). In the prior literature, many authors have defined a root cause as a target problem cause that the management has the power to control (Andersen and Fagerhaug 2006; Ammerman 1998; Rooney and Vanden Heuvel 2004; Livingstone, Jackson, and Priestley 2001). A root cause has also been defined as any underlying cause of the target problem (Rooney and Vanden Heuvel 2004). Additionally, a root cause has been defined as the deepest cause at the end of the causal structure (Andersen and Fagerhaug 2006; Ammerman 1998). However, this latter definition is contradictory due to the law of causality (see Section 2.1) as the “deepest causes” do not exist if everything that exists is caused by some earlier actions.

2.3 RCA in software process improvement

One goal of software process improvement (SPI) is to prevent software project failures. In order to reach this goal, SPI requires in-depth knowledge about the problems of past software projects (Boh, Slaughter, and Espinosa 2007; Edmondson 1996; Von Zedtwitz 2002). Such in-depth knowledge has been obtained from experienced individuals by using software project retrospectives (Dingsøyr 2005) at various levels, including the levels of teams (Birk, Dingsøyr, and Stålhane 2002; Bjørnson, Wang, and Arisholm 2009), organizations, and companies (Stålhane et al. 2003; Kalinowski, Travassos, and Card 2008). The team-level retrospectives are conducted by software teams and are aimed to analyse the problems relevant to the project goals of the teams. The organization-level retrospectives are conducted with participants representing the stakeholders of the whole software organization and are aimed to analyse the problems relevant to the project goals of the organization. The company-level retrospectives are conducted with participants representing the stakeholders of the whole software company, which aims to analyse the problems relevant to the company goals.

Figure 1 summarizes the flow of improvements from problematic software projects towards improved ones. The flow starts by recognizing the problems of past projects. It continues by using the team-, organization-, and company-level retrospectives analysing why the problems occurred, and it ends by controlling those problems in future projects. It presents the use of RCA as part of making improvements over the software projects, i.e. to explain why the problems of past projects occurred (Dingsøyr, Moe, and Nytrø 2001).

2.3.1 RCA of retrospective methods

In software project retrospectives, the use of RCA results in the creation of perceived causal models for the target problems (Stålhane et al. 2003). Software project retrospectives have utilized two RCA methods: 1) defect causal analysis (Bhandari et al. 1993) and 2) post-mortem review (Collier, DeMarco, and Fearey 1996). Both of these methods follow two work phases: 1) the detection of a target problem and 2) the detection of root causes. The RCA methods

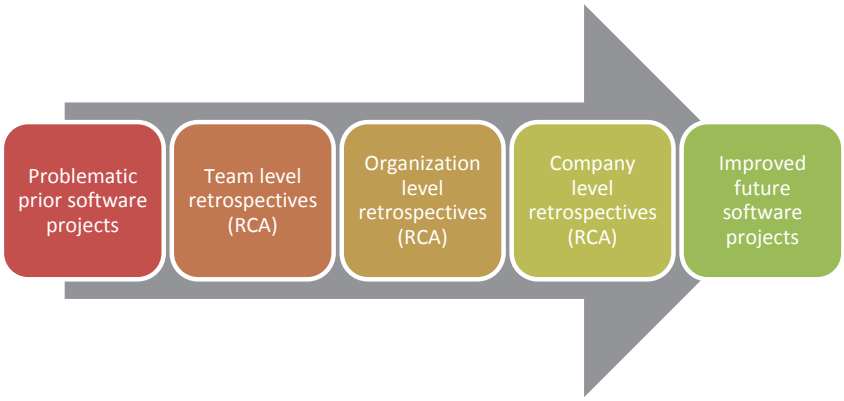

Figure 1. The flow of improvements from problematic software projects.

vary in terms of their aim and the work practices used in the work phase of the detection of a target problem. Instead, the methods follow similar work practices in the work phase of the detection of root causes. These two work phases are discussed below.

The detection of a target problem is the first phase of RCA methods, and its goal is to define the target problem for the second work phase. In defect causal analysis, the target problems include specific types of software defects. In a post-mortem review, the target problems may include any type of SE problems faced by individuals. Furthermore, the work practices of defect causal analysis utilize formal defect sampling combined with statistical methods including defect classifications and Pareto analysis (Card 1998). In comparison, the work practices of a post-mortem review are less formal and may include project surveys (Collier, DeMarco, and Fearey 1996) and brainstorming with individuals (Bjørnson, Wang, and Arisholm 2009). Furthermore, a post-mortem review also includes the detection of project success factors (Collier, DeMarco, and Fearey 1996), whereas defect causal analysis only detects problems that have occurred.

The detection of root causes is the second phase of RCA methods, and its goal is to explain why the target problems occurred. The RCA methods commonly use a retrospective meeting where a group of people analyse why the target problem occurred (Card 1998; Stålhane et al. 2003). There the detection of target problem causes is conducted by constantly asking "why?" for every cause of the target problem (Jalote and Agrawal 2005). Additionally, cause-effect diagrams are commonly used to organize and register the target problem causes based on their perceived causal relationships (Card 1998; Bjørnson, Wang, and Arisholm 2009; Stålhane 2004).

2.3.2 The environment of use

RCA has been used in the project retrospectives of small and large organizations. However, most prior studies on RCA have been conducted in large organizations (Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Gupta et al. 2008; Grady 1996; Mays 1990), as also noted by Stålhane et al. (2003). In large organizations, the optimal work practices for detecting and defining the target problems seem to be different than in small organizations. In large organizations, the target problems are detected with problem sampling (Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Gupta et al. 2008; Grady 1996; Mays 1990; Collier, DeMarco, and Fearey 1996). Instead, in small organizations, the target problems are detected by brainstorming (Dingsøyr, Moe, and Nytrø 2001; Stålhane et al. 2003).

A *lightweight RCA method* has been defined as an RCA method that can be conducted in a retrospective meeting lasting "half a day" (Dingsøyr, Moe, and Nytrø 2001). It seems that the use of brainstorming in the target problem detection phase makes the RCA method lightweight, as the detection of target problems can be conducted in the same retrospective meeting than the detection of root causes (Dingsøyr, Moe, and Nytrø 2001). Such an RCA method does not require heavy start-up investments and is adaptable to various target

problems. Regarding the prior literature, lightweight RCA methods are feasible for small companies (Stålhane et al. 2003), whereas large organizations require more effort in order to define the target problem for RCA (Card 1998). In large organizations, the target problem has to be well defined as otherwise the number of target problem causes is too high (Jalote and Agrawal 2005). Regarding the prior literature, the target problems of large organizations are often related to software defects. The target problems of small companies, on the other hand, have been related to high-level causes of software project failures, e.g. to estimation problems (Stålhane et al. 2003).

2.4 Gaps in the prior studies of RCA

There are three major gaps in the prior studies, which are considered in this thesis. These gaps are presented in the following sub-sections.

2.4.1 Work practices of RCA

Retrospectives should be lightweight, because otherwise they are neglected (Glass 2002). The concrete work practices of RCA are fairly little-studied in the context of software project retrospectives. There are only a few studies (Stålhane et al. 2003; Bjørnson, Wang, and Arisholm 2009; Dingsøyr 2005) on how to use RCA to detect, organize, and select the root causes of a software project failure. Additionally, the RCA methods presented by many authors are either heavyweight or too generally introduced to be adopted as such, e.g. Card (1998) introduces the mandatory phases of RCA but does not concretize how the phases are conducted. Additionally, most of the prior studies on RCA have been conducted in large organizations. Instead, an RCA approach could be useful for SME organizations too, but it has been rarely studied in such contexts (Stålhane et al. 2003).

Furthermore, the need to conduct RCA in distributed retrospectives has been introduced (Stålhane et al. 2003). However, there are no prior studies on how to use RCA in such circumstances. Software tools are used in distributed retrospectives to support real-time collaboration and information exchange over the distributed sites, but the prior tools (Terzakis 2011) do not enable the co-creation of a cause-effect diagram. Thus, conducting RCA in distributed retrospectives becomes difficult. The first research problem follows.

Research problem 1*: How can RCA be conducted in collocated and distributed software project retrospectives?*

2.4.2 Perceptions of practitioners

RCA has been presented as a feasible approach to software project retrospectives. Thus, it could be useful for software process improvement. However, its perceived ease of use, cost-efficiency, and added value have not been widely studied.

There are only a few studies that have compared the use of RCA with the retrospectives which do not use it (Dingsøyr, Moe, and Nytrø 2001; Card 1998; Stålhane et al. 2003; Stålhane 2004). Similarly, the RCA methods are not widely compared with one another (Stålhane et al. 2003; Bjørnson, Wang, and Arisholm 2009). The effort required to conduct RCA (Card 1998; Grady 1996; Mays 1990) has also been neglected in most of the studies. Furthermore, how participants experience RCA (Birk, Dingsøyr, and Stålhane 2002) is not widely studied, i.e. do the software developers experience RCA as a useful approach for lightweight retrospectives?

Although the prior studies of RCA are promising (Card 1998; Leszak, Perry, and Stoll 2000), they do not indicate whether RCA is useful in retrospectives in which problems other than technical quality deviations are analysed, e.g. Stålhane et al. (2003). The second research problem follows.

***Research problem 2**: Is RCA perceived as efficient and easy to use in software project retrospectives?*

2.4.3 Outcome of RCA

RCA takes a problem as an input and provides the perceived causal relationships as an output. Thus, theoretically, RCA could be a feasible approach to explaining why a software project failed. It could reveal not only what happened and where it happened, but also why it happened—a gap in prior studies on software project failures, discussed further in Article IV. In practice, however, RCA has not been widely reported as being feasible for such purposes. There are only a few real-world studies (Stålhane et al. 2003) indicating that RCA reveals any interconnections between the causes of software project failures. Instead, most of the industrial cases of RCA (e.g. Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Gupta et al. 2008; Grady 1996) have studied the causal relationships between target problems and their individual causes only while disregarding the analyses of their mutual causal relationships. Thus, the real-world case studies on the use of RCA to explain the perceived causal relationships of software project failures are scarce. The third research problem follows.

***Research problem 3**: Does the outcome of RCA indicate how the causes of software project failures are interconnected?*

3. Research approach and methodology

The main objective of this thesis is to develop and evaluate a lightweight RCA method (called ARCA) and software tool (called ARCA-tool) for the software project retrospectives of SME organizations. The overall research approach is design science (Hevner et al. 2004). The approach includes artefact development and empirical evaluation with the mixed-methods approach (Shull, Sjøberg, and Singer 2008), that combines three main research approaches: observation-based industrial field studies (Lethbridge, Elliott Sim, and Singer 2005), case studies (Yin 1994), and controlled experiments (Juristo and Moreno 2003).

This section starts by introducing the research questions (RQ) that are aimed at contributing to the research problems (see Section 2.4). Thereafter, Section 3.2 presents the research articles (I-V). Section 3.3 introduces the framework of design science and Sections 3.4 to 3.7 present the use of the framework in the development and evaluation of the ARCA method and ARCA-tool. Figure 2 summarizes the linkages between the research problems, the research questions, and the studies of the thesis.

3.1 Research questions

The development and evaluation of the ARCA method and ARCA-tool answers a total of seven research questions. These are introduced below.

3.1.1 Development of the ARCA method and ARCA-tool

There are two research questions about the first research problem, namely, "*How can RCA be conducted in collocated and distributed software project retrospectives?*" The studies of this thesis focused on creating knowledge about the environment of use and the literature of RCA methods (Article I) and RCA software tools (Article III).

Research question 1: *What are the common steps of RCA methods, and how are they to be conducted?*
The first research question reviews prior RCA methods and synthetizes their commonalities and work practices. This knowledge is thereafter used to develop the ARCA method and ARCA-tool.

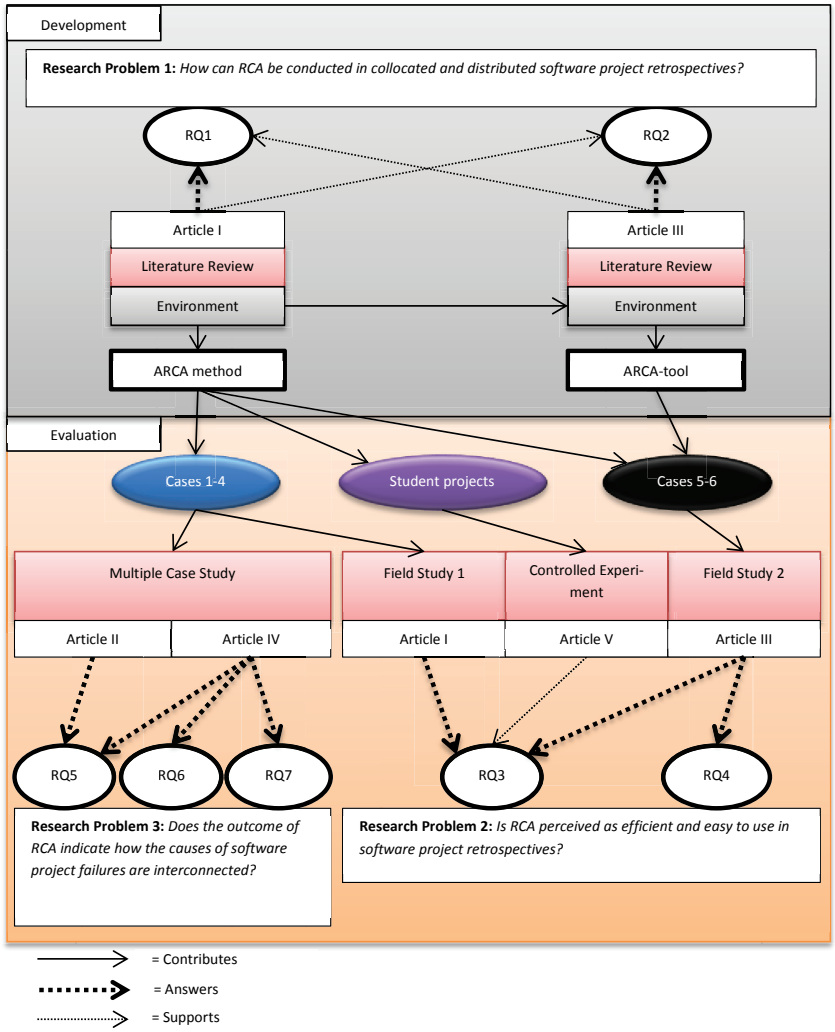

Figure 2. The summary of the research approach linking the research problems, research questions, and studies of the thesis.

Research question 2: *What software tools for RCA are introduced, and how do they support software project retrospectives?*

The second research question considers alternative software tools for RCA. The research question reviews prior RCA software tools and compares their main features for conducting RCA in collocated and distributed software project retrospectives.

3.1.2 Ease of use and cost-efficiency evaluations

There are two research questions about the second research problem, "*Is RCA perceived as efficient and easy to use in software project retrospectives?*" The studies of this thesis include two industrial field studies introduced in articles I (Cases 1-4) and III (Cases 5-6). Additionally, a controlled student experiment was conducted (Article V).

Research question 3: *Is the ARCA method perceived as efficient and easy to use for analysing software engineering problems in software project retrospectives?*
The third research question evaluates the perceptions of practitioners on the ARCA method. The evaluation is limited to usefulness, ease of use, and the ARCA method outcome. The use of the method covers collocated and distributed software project retrospectives.

Research question 4: *Is the developed ARCA-tool perceived as useful and easy to use in software project retrospectives applying the developed RCA method?*
The fourth research question evaluates the perceptions of practitioners using ARCA-tool. The evaluation is limited to usefulness and ease of use of the tool in industrial retrospectives.

3.1.3 Outcome of RCA with software project failures

There are three research questions about the third research problem, "*Does the outcome of RCA indicate how the causes of software project failures are interconnected?*" The main assumption regarding the research problem is that the outcome of RCA should express what and where the causes of failures occur and how they are interconnected over the process areas. The research data is based on the same cases (Cases 1-4) that were used in Field Study 1 (Article I), as can be seen from Figure 2.

Research question 5: *Which process areas and cause types were frequently used in RCA to explain software project failures?*
The fifth research question expresses what the perceived causes of failures were and where in the development process they occurred. Taxonomy of the perceived causes was developed, evaluated, and applied in order to answer this research question.

Research question 6: *What causal relationships bridge the process areas?*
The sixth research question considers how the process areas were interconnected in the cases of software project failures. A *bridge cause* refers to a detected cause of failure for which the process area of the effect is different from the one of the cause. The thesis includes qualitative analyses of bridge causes.

Research question 7: *Do the causes perceived as feasible targets for process improvement differ from the other detected causes, and if so, how?*
The seventh research question considers the role of cause types, process areas, and bridge causes for process improvement activities. The thesis analyses the perceptions of practitioners and senior management on the detected causes that are feasible targets for process improvement.

3.2 Research articles

There are a total of five research articles in this thesis (see Figure 2). Articles I and III include the development of the ARCA method and ARCA-tool. Articles

I, III, and V include the perceived ease of use and cost-efficiency evaluations. Articles II and IV include the analyses of the ARCA method outcome.

Article I answers RQ1 and RQ3. It reviews and synthetizes prior RCA methods and their work practices. Article I also presents the development of the ARCA method. The method is evaluated in four industrial cases (Cases 1-4) aimed to reveal the causes of software project failures.

Article III answers RQ2-RQ4. It reviews and compares prior RCA software tools. Article III also presents ARCA-tool. Article introduces how the tool supports the ARCA method. The software tool and the ARCA method are further evaluated in two industrial cases (Cases 5-6).

Article V contributes to RQ3. It considers the actual and perceived effect of using a cause-effect diagram in the RCA of software project retrospectives. The article compares the use of the cause-effect diagram of the ARCA method and ARCA-tool with an approach of writing down simple memos during RCA. Such a comparison was important to conduct in order to separate the effect of the cause-effect diagram from the structured investigation of ARCA.

Article II answers RQ5. It presents general cause types and process areas, explaining where in the development processes the causes of software project failures occur. The outcome of the ARCA method in Cases 1-4 was used in the analysis.

Article IV answers RQ5-RQ7. It extends Article II by including in-depth analyses of the cause types, process areas, bridge causes, and feasible targets for process improvement in the case companies. Article IV presents that in a case of software project failure, the ARCA method helps to express what happens, where it happens, and why it happens.

3.3 The framework of design science

The development and evaluation of the ARCA method and ARCA-tool were conducted by using the framework of design science (Hevner et al. 2004; March and Smith 1995). The framework (see Figure 3) consists of the environment, knowledge base, and artefact design (Hevner et al. 2004). In this thesis, the artefact design includes the development and evaluation of the ARCA method and ARCA-tool.

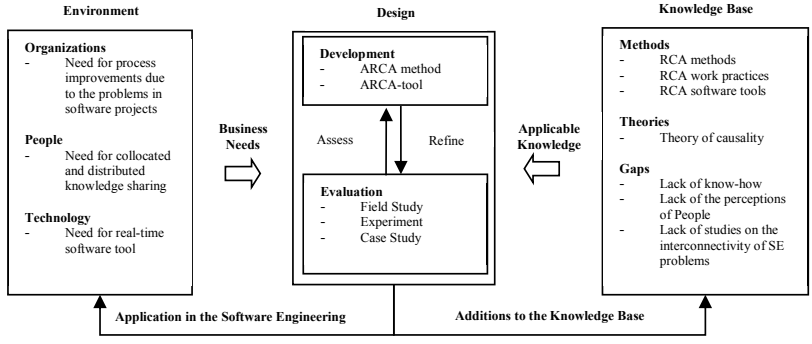

Figure 3. Framework of design science (Hevner et al. 2004).

3.3.1 The environment

Environment refers to the context in which the ARCA method and ARCA-tool were planned to be used. The environment was *small- and medium-sized* international software product companies that needed a lightweight problem-prevention method for their distributed software organizations and individual software teams. In order to understand the environment, business needs in the problem-prevention method were considered in six software product companies (Cases 1-6).

The business needs resulted in the requirements of the ARCA method (see Section 3.4.1) and ARCA-tool (see Section 3.4.2). The first business need was related to the problems of software projects. The companies had faced major problems in their software development projects due to the complex products (Article IV). The problems included product quality issues and the schedule and effort overruns of the projects. The existing problem-prevention methods of the companies were feasible for detecting problems that had occurred but infeasible for conducting in-depth analyses of the causes of problems (articles I and III).

The second business need was related to the needs of distributed software development. There were problems of facilitating lightweight software project retrospectives due to the distributed software development teams (Article III). Arranging face-to-face retrospectives with geographically distributed team members required too much effort. Respectively, a lack of real-time software tool support made it difficult to conduct distributed retrospectives.

3.3.2 The knowledge base

The knowledge base was the prior literature on problem prevention theories, methods, work practices, and software tools. It revealed that the literature on RCA was relevant to problem prevention and software project retrospectives. Therefore, the applicable knowledge on RCA methods, work practices, theories, and software tools was gathered. The knowledge was used to develop the ARCA method and ARCA-tool.

3.3.3 The artefact design

The business needs of the environment combined with the applicable knowledge were used to design the ARCA method and ARCA-tool. The development of these artefacts was important because the applicable knowledge did not include feasible solutions in the environment. For example, defect causal analysis would have required too much effort, and it would have been adaptable to software defects only (Article I). Respectively, post-mortem review would have been infeasible for geographically distributed organizations (Stålhane et al. 2003), and it would have been adaptable to project experiences only. Additionally, the applicable knowledge revealed gaps in the prior studies (see Section 2.4), which made the development and evaluation of the ARCA method and ARCA-tool also scientifically interesting. Therefore, the development of the ARCA method and ARCA-tool became reasonable. The companies

needed a method for distributed organizations useful for understanding the causes of software project problems, and we wanted to know how to conduct RCA with SME organizations over the collocated and distributed retrospective settings.

3.4 Development of the ARCA method and ARCA-tool

The ARCA method and ARCA-tool are introduced in Sections 4 and 5. This section introduces how the ARCA method and ARCA-tool were developed, which provided knowledge relevant for the research questions RQ1 and RQ2.

3.4.1 Development of the ARCA method

The development of the ARCA method was initialized by setting down its requirements. Based on our understanding of the environment (see Section 3.3.1), we started by brainstorming the characteristics of a "beneficial" RCA method for software companies. We concluded that such a method would help the companies to develop high-quality corrective actions with low effort. This conclusion resulted in the following requirements:

1. The method helps to develop feasible and effective improvements
2. The method requires low effort
3. The method is easy to use
4. The method is adaptable to different kinds of target problems

Thereafter, a literature review was conducted. The review covered RCA methods introduced in industrial engineering contexts. The search was limited to the literature found in Google and Scopus. The following search words were used to find the relevant literature: "RCA", "root cause analysis", "DCA", "defect causal analysis", "defect analysis", "defect prevention", and "problem prevention". The review answered the following research questions introduced in Article I:

1. Are there steps common to RCA methods?
2. What are the recommended work practices in the different steps of RCA?

Then, the first version of the ARCA method was created. It was based on the requirements and the findings from the literature review. Analytical argumentation for alternative work practices was used to develop the method. The first version of the ARCA method was piloted with a student software project (Article I).

3.4.2 Development of ARCA-tool

The development and evaluation of the ARCA method revealed the need for using a software tool during RCA. For example, collaborative cause-and-effect diagramming and idea development was found to be important in distributed

retrospectives. Unfortunately, the existing RCA software tools were infeasible for the environment of use (Article III). Therefore, an RCA software tool, named ARCA-tool, was developed.

ARCA-tool was developed in two subsequent projects on the Aalto University software capstone project course[1]. During the projects, the author of this thesis acted as the customer and provided the tool requirements. The software tool was designed to be used in the synchronous retrospective meetings of small software project teams including a maximum of ten team members. Additionally, the tool was required to support collocated and distributed software project retrospectives. The tool was also required to be simple and easy to use. The main requirements included the following (Article III):

1. Supports real-time collaboration over distributed team members
2. Enables co-creation of a cause-effect diagram
3. Enables developing ideas for the causes of problems
4. Enables voting for the most severe causes and best ideas
5. Enables capturing and refining the outcome of retrospectives
6. Protects the anonymity of team members
7. Is simple and easy to use

3.5 Field study evaluations

Field studies are commonly used to improve and understand real-life work practices and tools (Lethbridge, Elliott Sim, and Singer 2005). Therefore, they were useful for evaluating the ARCA method and ARCA-tool. The observation-based industrial field studies with software product companies were used to evaluate the perceptions of practitioners using the ARCA method and ARCA-tool. Six industrial cases (Cases 1-6) were conducted, and they were used to answer the research questions RQ3 and RQ4 (see Figure 2). Table 1 summarizes these cases.

The field studies were positioned to cover incremental and agile software development approaches. Additionally, they were positioned to cover distributed software development settings. Furthermore, the field studies were positioned to cover retrospectives at various levels of analysis, including the levels of company, organization, and team (see Section 2.3).

Regarding data collection, the ARCA method was evaluated by the participating people, i.e. the employees were interviewed and asked to provide feedback with questionnaires. The participants compared the ARCA method with the existing practices of the companies. The cases were video recorded and observed.

The cases varied. First, in Case 5, the participants had previously used the ARCA method and ARCA-tool. Instead, in Cases 1, 2, 3, 4, and 6, the ARCA method and ARCA-tool were not used previously. The existing practices

[1] https://noppa.aalto.fi/noppa/kurssi/t-76.4115/etusivu

Table 1. The summary of the field study cases.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Organization	SME	SME	SME	SME	SME	SME
Evaluation domain	Product company	Product company	Product company	Product company	Product organization	Software team
SW Process	Incremental	Incremental	Incremental	-	Scrum	Scrum
Retrospective	1 x ARCA	1 x ARCA	1 x ARCA	1 x ARCA	3 x ARCA	1 x ARCA
Software tools	Diagram + monitor	Diagram + monitor	Diagram + monitor	Diagram + monitor	ARCA-tool + monitor	ARCA-tool only
Participants	Various stakeholders (N=9)	Mostly developers (N=9)	Various stakeholders (N=7)	Various stakeholders (N=6)	Various stakeholders (N=11)	Mostly developers (N=5)
Focus	Defects	Defects	Installation	Lead-time	Requirements	Team level
RCA experiences	No RCA	5-whys	No RCA	No RCA	Post-mortem reviews	No RCA

included “discussions about problems”, followed by the development of corrective actions. Neither RCA nor cause-effect diagrams were widely used previously. Case 2 had “tried” 5-whys approach (Article I).

Second, in Cases 1-4, the ARCA method was focused on problems at the company level, i.e. high-level problems that caused software project failures. Instead, in Cases 5-6, the ARCA method was applied to more focused problems. These included a problem of weak requirement specifications in a software organization (Case 5) and work-practice problems of an individual software team (Case 6).

Third, Cases 1-4 were conducted at software organizations with more traditional, incremental software development processes. Instead, Cases 5-6 were conducted at software organizations with modern agile software development processes.

Fourth, in Cases 1-5, the ARCA method was conducted collocated. Instead, in Case 6, the ARCA method was conducted distributed, filling the gap of limiting the evaluations to collocated settings only.

Fifth, only Cases 5 and 6 evaluated ARCA-tool. The tool was developed after Cases 1-4. Therefore, they were not used to evaluate the tool. Instead, Cases 1-4 helped to consider the requirements of the tool.

3.5.1 Field studies at Cases 1-4

Cases 1-4 (Article I) were conducted in four medium-sized software organizations. The rationale for the selection of the case sites was that together they allowed us to evaluate the ARCA method in different software engineering contexts where RCA has not been used previously. Thus, the evaluation provides rich information about the improvement over the existing practices.

In the data collection, the data sources and the data collection methods were triangulated in order to increase the reliability of the results (Yin 1994; Rune-

son and Höst 2008; Jick 1979). We used interviews (Yin 1994), questionnaires (Foddy 1994), measurements, and observations (Yin 1994) to evaluate the perceived usefulness and ease of use of the ARCA method. A total of five key representatives of the companies were interviewed, and 30 participants answered the questionnaires.

Interviews were conducted with *key representatives*. Key representatives were company managers involved in steering their RCA case who had the power to make process changes in their companies. The interviews were held before and after a company case in order to analyse how the key representatives experienced the ARCA method. The interview questions were tested with researchers before the company cases.

The questionnaires were used after the two main steps of the ARCA method, namely *root cause detection* and *corrective action innovation*, in order to analyse how the case participants experienced the ARCA method and its output. Closed- and open-ended questions were included in the questionnaires, as recommended by Foddy (1994). The interval between the questionnaire items was equal. The scale in each item was symmetric (1 = very low; 2, 3, 4 = neutral; 5, 6, 7 = very high). The questionnaires were tested by researchers and students before using them.

The effort used and the output of the ARCA method was measured in each case. An accurate record of the used man-hours in each step of the ARCA method was kept. Additionally, we registered the number of detected and processed causes of target problems in each case. We also registered the number of developed corrective actions and the evaluations of participants regarding the perceived feasibility and impact of each corrective action.

Observations were conducted by two researchers. One steered the ARCA method together with the key representatives, whereas one observed the actions of the ARCA method. Both researchers wrote notes during the case. The researchers held a feedback session after the ARCA method steps of root cause detection and corrective action innovation. The observations were used to consolidate the results from the interviews and questionnaires.

The data analysis was conducted in two phases. First, after each company case, we considered the collected research data to conclude the strengths and weaknesses of the ARCA method. Second, after all company cases were conducted, we evaluated the ARCA method as a whole by combining all empirical evidence from the company cases.

3.5.2 Field studies at Cases 5-6

Cases 5-6 (Article III) were conducted in two organizations: a medium-sized and a small-sized software development organization. The rationale for the selection of these two case sites was that together they enabled us to evaluate the ARCA method and ARCA-tool in collocated and distributed agile software engineering contexts. Unlike in Cases 1-4, the company personnel steered the use of the ARCA method.

The ARCA method was limited in Cases 5-6. The work phases of *preliminary cause collection* and *corrective action workshop* (see Section 4.2) were ex-

cluded from the cases. Therefore, the evaluation results regarding the ARCA method are also limited, respectively. For the convenience of the reader, the ARCA method is called the *limited ARCA method* when referring to the use of the ARCA method without these work phases.

Similarly to the first field studies, the data sources and research methods were triangulated. We used interviews (Yin 1994), questionnaires (Foddy 1994), and observations (Yin 1994) to evaluate the perceived usefulness and ease of use of the limited ARCA method and ARCA-tool. A total of 16 case participants filled in the questionnaires and a total of eight participants were interviewed. The scale of questionnaire items was symmetric: 1=very minor, 2, 3, 4, 5=very major (Case 5) and 1=very low, 2, 3, 4, 5=very high (Case 6).

During the data analysis, the interviews and questionnaires were summarized in order to conclude whether the perceptions of participants were similar between the cases. Both cases were first analysed separately, because the questionnaires and interviews varied slightly between the cases. This was due to differences in the company contexts. Case 5 had used the limited ARCA method and ARCA-tool previously, while Case 6 had not. After the interviews were conducted, we transcribed and coded them accordingly. After the analysis of both cases, we summarized the results from both cases in order to compare their similarities and differences.

3.6 Controlled experiment evaluations

Controlled experiments are commonly used to compare alternative work practices and tools, e.g. Bjørnson et al. (2009). In this thesis, a controlled experiment was used to extend the field studies by focusing on the cause-effect diagram of the ARCA method and ARCA-tool. The experiment with 11 student software project teams (61 participants) was conducted to evaluate the impact of using the cause-effect diagram with the limited ARCA method. The data collection considered the perceptions of participants and the outcome of the limited ARCA method (see Section 3.5.2). The experiment results provide additional evidence for the research question RQ3 (see Figure 2).

3.6.1 Research context

The experiment was conducted with software project teams of a capstone project course in Aalto University. In the course, students develop real-world software for real-world customers in teams. Each software project lasts five months. The challenges encountered by the project teams are close to the challenges encountered in industrial software development.

Each team includes seven to nine student members divided into roles: three managers and four to six software developers. Additionally, each team follows an iterative process framework, which is defined by the course. The framework divides the projects into three time-boxed iterations, each lasting six to seven weeks.

The experiment was conducted in the retrospectives of eleven project teams out of fourteen during the academic year 2010-2011. The participation in the

experiment was voluntary for the project teams. Table 2 summarizes the retrospectives of the teams divided into the techniques used to organize the causes of problems during RCA. The table presents the main focus of the retrospectives. We can see that most of the teams focused on similar target problems in both retrospectives. The number of participants and the language used remained similar over the retrospectives.

The use of students as study subjects has been discussed in the SE literature (Svahnberg, Aurum, and Wohlin 2008; Berander 2004; Carver et al. 2003; Runeson 2003; Höst, Regnell, and Wohlin 2000). The student subjects of the controlled experiment were graduate-level students, who were experienced in software engineering and motivated to reach up to their project goals. Thus, they were feasible targets for revealing the trend of improvements (Berander 2004; Runeson 2003). Additionally, the student projects were close to "real software projects". Thus, also the challenges faced by the students were industrially relevant, as we concluded in Vanhanen et al. (2012).

3.6.2 Experiment design

The author of this thesis controlled the methods and settings of each retrospective. As required by the course framework, each team conducted retrospectives at the end of the second and third iteration. Thus, the experiment design was limited to two experimental units for each team, 22 experimental units as a total. The retrospective method and the used effort were fixed for each unit.

The experiment was conducted by using a single factor paired design including one blocking variable (Juristo and Moreno 2003). The examined factor

Table 2. The summary of the retrospectives.

Team	**Ltd. ARCA method** (Cause-effect diagram)				**Control Group** (List-of-causes)			
	#	L	Target problem	∑p ∑c c/p	#	L	Target problem	∑p ∑c c/p
1	1	F	Co-operation, management	5 76 15	2	F	Co-operation, management	4 70 18
2	1	F	Scope, quality	7 87 15	2	F	Quality, scope	6 59 10
3	2	E	Scope, development	5 93 19	1	E	Co-operation, management	6 78 13
4	1	F	Scope, quality	6 127 21	2	F	Quality, scope	5 85 17
5	1	F	Co-operation, customer	6 137 23	2	F	Quality, customer	6 92 15
6	1	F	Tasks, motivation	5 121 24	2	F	Motivation, skills	5 137 27
7	2	F	Scope, task monitoring	5 111 22	1	F	Task monitoring, scope	6 98 16
8	2	E	Process, skills	6 109 18	1	E	Process, skills	6 97 16
9	2	F	Management, co-operation	5 129 26	1	F	Co-operation, management	5 125 25
10	1	E	Requirements, risk management	6 69 12	2	E	Requirements, skills	6 90 15
11	2	F	Co-operation, management	5 113 23	1	F	Co-operation, management	6 10 17
			Mean	6 107 20			Mean	6 94 17

#=indicates whether the technique was used in the first (1) or second (2) retrospective, L=used language (F=Finnish, E=English), ∑p=the number of participants, ∑c=the number of detected causes, c/p=the average number of detected causes per participant

was the technique used to visualize the causes of problems. The factor had two alternatives, including the cause-effect diagram of the ARCA method (see Figure 4) and a list-of-causes (see Figure 5). Considering the main differences between the alternatives, arrows are drawn between the causes of the problem when using the cause-effect diagram. Instead, in the list-of-causes, there are no arrows between the causes of the problem; the causal structure is visualized by using bulleted lists. Furthermore, in the case of many effects being caused by one cause, we can see that multiple arrows can be drawn from a cause under the related effects with the cause-effect diagram. Instead, with the list-of-causes, such cause needs to be duplicated under each effect.

Both alternatives of the examined factor were used in each team, but in different retrospectives. The project phase created a blocking variable that could not be fully eliminated. The experiment design was balanced by 1) randomizing the starting order of the alternatives for each team and 2) forcing half of the teams to start with the cause-effect diagram and the rest with the list-of-causes technique. Additionally, paired analysis between the alternatives inside each team was used to compare the differences, which mitigated differences between teams. Table 3 summarizes the balanced design, including the distribution of teams in the alternatives and the related project phase when used.

Table 3. Distribution of alternatives (A=Cause-effect diagram, B=List-of-causes) into 22 units (Article V).

		Team										
		T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
Phase	I2	A	A	B	A	A	A	B	B	B	A	B
	I3	B	B	A	B	B	B	A	A	A	B	A

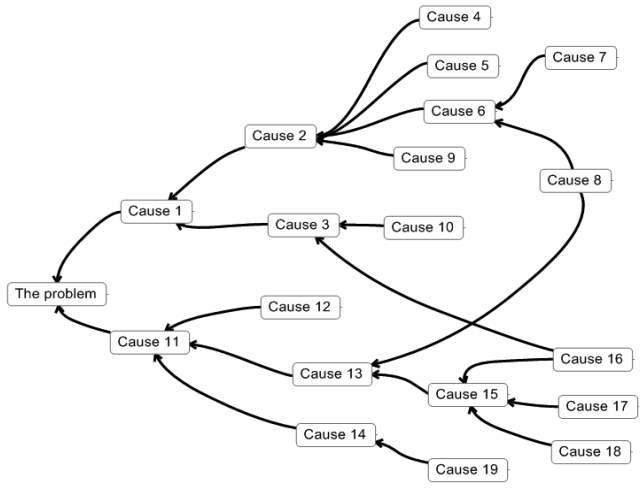

Figure 4. The cause-effect diagram used in the A alternative (Article V).

The Problem
- Cause 1
  - Cause 2
    - Cause 4
    - Cause 5
    - Cause 6
      - Cause 7
      - Cause 8
    - Cause 9
  - Cause 3
    - Cause 10
    - Cause 16
- Cause 11
  - Cause 12
  - Cause 13
    - Cause 8
    - Cause 15
      - Cause 16
      - Cause 17
      - Cause 18
  - Cause 14
    - Cause 19

Figure 5. The list-of-causes used in the B alternative (Article V).

3.6.3 Response variables and research hypothesis

There were five response variables that were compared over the alternatives of the examined factor. These included *Method Effectiveness* (ME), *Size of Depth Levels* (SoDL), *Number of Hub Causes* (NoH), *Characteristics of Detected Causes* (CDC), and *Perceptions of Participants* (PP).

The efficiency of the retrospective method has been measured with the number of detected causes (Bjørnson, Wang, and Arisholm 2009). The response variable ME indicates the number of unique problem causes detected. According to our hypothesis, using the cause-effect diagram results in a higher ME than using the list-of-causes.

Causal Structure is related to the causal structure of the causes of the problem. Regarding Causal Structure, we recognized the response variables SoDL (Bjørnson, Wang, and Arisholm 2009) and NoH (Bjørnson, Wang, and Arisholm 2009). The response variable SoDL indicates the number of causes over different Depth levels, defined as the number of cause-effect pairs from a cause to the target problem. A function SoDL(x) was created to measure SoDL. The function returns the number of causes being registered to Depth level x. We hypothesized that with both alternatives, the return value of SoDL(x) increases among the Depth levels, but the return values of SoDL(x) are larger with the cause-effect diagram. The response variable NoH indicates the number of "hub causes", defined as a cause which explains more than one cause. NoH was measured by calculating the number of effects stemming from each cause. We hypothesized that NoH is a higher number with the cause-effect diagram. The cause-effect diagram of the ARCA method is a directed graph, whereas the list-of-causes is a tree. A tree-structured cause-effect diagram has been compared with the directed graph-structured cause-effect diagram (Bjørnson, Wang, and Arisholm 2009). The prior study indicates that the directed graph-structured cause-effect diagram results in increasing NoH values (Bjørnson, Wang, and Arisholm 2009).

Third, we also assumed that the technique used to visualize the causes of problems did not artificially steer the discussions of retrospectives into different types of problems or causes. Instead, it was possible that the project domain including changing situations could affect the causes of problems detected in the retrospectives. CDC was used to measure the differences in the discussion contents of retrospectives. We hypothesized that there is no difference in CDC over the alternatives. CDC was measured for each retrospective by using a classification system, which characterizes the types and process areas of failure causes (Article IV). During the data analysis, the distributions of causes in cause classes over the alternatives were compared by using linear correlations.

Fourth, we measured PP, which indicates how the participants perceived the alternatives. The prior literature has commonly recommended using the cause-effect diagram in RCA. Thus, we assumed that the participants prefer using it in retrospectives. In order to measure PP, a questionnaire (see Article V) was used after each retrospective. The answers of participants who were not involved in both retrospectives (10 of 61 participants) were excluded. Additionally, after both retrospectives were conducted, another questionnaire combined with a group interview was used to compare the alternatives.

3.6.4 Controlling undesired variation

Learning effect and team specific contextual factors likely affected the outcome of the retrospectives. We were not able to eliminate the blocking variable related to the project phase. Therefore, it was important to ensure that the contextual factors were similar in each experimental unit. A total of six context variables were controlled. These included the high-level goal of retrospectives, the number and roles of participants, the used language, the physical context, and the retrospective facilitator. Additionally, we identified and measured three confounding variables, since we had no control organizing the course's project teams and their customer's topics. The confounding variables included the specific target problem of the retrospectives (see Table 2), team members' motivation, and team spirit.

3.6.5 Data analysis

We used the outcome of the retrospectives in statistical analyses on ME, SoDL, NoH, and CDC. In order to analyse PP, we combined statistical methods with qualitative methods.

ME was analysed with the paired-samples two-tailed t-test with alpha level 0.05. The tests were conducted for the total number of detected causes and for the average number of detected causes per participants.

SoDL and NoH were also compared by using the paired-samples two-tailed t-test with alpha level 0.05. Over the retrospectives of each team, we analysed whether the cause-effect diagram results systematically in larger SoDL(x) and NoH values than the corresponding list-of-causes technique.

CDC was analysed in order to show that the alternatives did not significantly affect the discussion contents of the retrospectives. We started the analysis by classifying the detected causes into type and process area categories (see Article IV). For each cause, the process area and cause type classifications were combined, which resulted in *a characteristic of the cause* (there were a total of 84 possible characteristics). After the characteristics were determined for each cause, the Pearson's correlation between the numbers of causes with the same characteristic was calculated over the retrospectives of the limited ARCA method and the control group. The correlation was calculated between the retrospectives of each team and between the retrospectives of all teams combined together. The closer the correlation is to 1, the less different were the discussion contents of retrospectives using the different alternatives.

The analyses of PP were based on questionnaires and group interviews. Questionnaire 1 was used after each retrospective to evaluate the work practices of the used retrospective method (Wilcoxon Signed Rank Test, alpha=0.05). Questionnaire 2 was used after both retrospectives were conducted. It was used to compare the retrospectives. The group interview was conducted after Questionnaire 2. It was used to understand the perceptions of participants.

3.7 Case study evaluations

The case studies are commonly used to understand real-life phenomena and events (Yin 1994). In this thesis, a multiple case study approach was used to evaluate the outcome of the ARCA method in industrial settings, aiming to explain the relationships between the causes of software project failures (articles II and IV). The results are used to answer the research questions RQ5-RQ7 (see Figure 2). The data analysis covered the perceived causes of software project failures and their perceived causal relationships in Cases 1-4. The selection of these case sites was reasonable, as together they allowed us to analyse the commonalities of the perceived causes of four different software project failures.

3.7.1 Data collection

The ARCA method was used as the data collection method at each case. Its detailed description can be found in Section 4.2. Therefore, this section only introduces the main phases and contextual settings of the ARCA method relevant to Cases 1-4.

Each case started with a focus group with senior managers who had the power to make process changes in their companies. The aim was to determine a high-level target problem that had caused project failures systemically. Measurable evidence was used in the focus group to testify to the occurrence of the target problem. Additionally, the senior managers selected company experts, nine people as an average, who should participate in the detection of the causes of the target problem. The company experts included people from various process areas covering sales & requirements, management work, software development, software testing, and release & deployment.

Following the focus group, the researchers arranged two work phases. These included a preliminary cause collection and causal analysis workshop. In the preliminary cause collection, the company experts were asked to provide at least five causes explaining why the target problem occurred. The preliminary cause collection was confidential for the experts, and it was conducted by an email exchange between the author of this thesis and each individual expert. Based on the preliminary cause collection, the researchers and senior managers created a cause-effect diagram. The managers analysed the diagram carefully and selected cause entities that should be analysed in the causal analysis workshop (see Section 4.2 for further details about the cause entities).

The causal analysis workshop was a time-boxed meeting of 120 minutes, which was conducted with the named company experts. During the meeting, new causes were detected under each selected cause entity. The meeting resulted in a finalized cause-effect diagram. It was used to explain why the software project failure occurred. Thereafter, the experts were asked to propose causes that they perceived as important to be further processed in the process improvement activities. Then the senior managers considered the diagram and made the final selection about the causes that were processed in the process improvement activities.

Considering the validity of the collected research data, we should note that it is based on the perceptions of people. The correctness and accuracy of the detected causes were evaluated in each case by the author of this thesis. Triangulation of the data sources and the data collection methods (Yin 1994; Runeson and Höst 2008; Jick 1979) increases the reliability of the detected causes. Before the preliminary cause collection was conducted, interviews were kept with the senior managers to detect the causes of failures which they perceived to be important. I assume that the causes they underlined in the interviews would also be recognized by the experts in the causal analysis workshop. The detected causes from both of these two groups were compared and it was found that in each case, the experts detected and extended most of the causes underlined by the senior managers. This comparison is documented in detail in Article IV as a part of the case study results. Additionally, interviews and questionnaires were used to evaluate the outcome of the ARCA method, which included the evaluation of the correctness and accuracy of the detected causes. Regarding these results, the experts and senior managers perceived that the detected causes were correct and accurate. This validation is presented later as a part of the field study results (see Section 6.1).

3.7.2 Data analysis

The data analysis included three phases. It started by analysing the types of causes and the related process areas expressing where the causes occurred (articles II and IV). Thereafter, it continued by analysing how the causes were interconnected (Article IV). Finally, the feasibility of causes for process improvement was studied (Article IV).

The data analysis was initialized by developing a detailed classification system. Thereafter, the system was applied to the ARCA method outcome. The

development of the classification system was iterative. First, a literature review was conducted. The literature review covered problem cause classification dimensions used in the software engineering context. The dimensions of process areas and cause types were concluded to be important. The dimension of process area expresses where in the development processes the cause occurs (Grady 1996; Dye and van der Schaaf 2002; Jacobs et al. 2005; Nakashima et al. 1999), and the dimension of cause type describes what the cause is, e.g. an issue in the product (Grady 1996; Dye and van der Schaaf 2002; Nakashima et al. 1999) or in the people (Leszak, Perry, and Stoll 2000; Stålhane 2004; Dye and van der Schaaf 2002; Jacobs et al. 2005). Followed by the literature review, preliminary categories for the dimensions of process areas and cause types were created. Thereafter, the preliminary categories were combined with an approach similar to the grounded theory, as suggested in Salinger et al. (2007). The author of this thesis classified a sample of causes from each case and simultaneously refined the preliminary categories to correspond better to the causes of our cases. After the classification dimensions were finalized, they were applied to all detected causes and their distributions were used to introduce what the problem causes of software projects were and where they occurred, introduced in Article II.

The analysis was continued by extending the work of Article II to individual cases combined with cross-case analysis, introduced in Article IV. During the analysis, the classification system was also slightly improved. For example, based on the results of inter-rater agreement (see Section 7.5.4), two process areas were combined together. The process areas of Development Work and Change Management were combined under the process area of Implementation. Respectively, some sub-categories were re-named, e.g. the sub-category Customers was re-named into Customers & Users. Additionally, some cause statements were excluded from the analysis, as the more detailed analysis showed that they were not the "real" causes of failures, but some coarse-grained statements about speculations given in the discussions of the causal analysis workshop, e.g. there was a cause statement: "There is a study from the States which concluded that software quality should be the most important goal for companies." The total number of excluded statements was 18 from a total of 648 statements.

The continued analyses also covered an analysis of the interconnectedness between the causes of project failures (Article IV). A new term, "bridge cause", was founded, which refers to a cause that links process areas together. The bridge causes were analysed qualitatively. The analysis was initialized by selecting the perceived causal relationships for which the cause and effect were classified in different process areas. Thereafter, the selected pairs of causes and effects were grouped according to their process areas. For each group, the perceived causal relationships were explored by considering the original cause-effect diagrams. The explored parts of the cause-effect diagrams were summarized and concretized in order to conclude how the causes and effects were interconnected over the related process areas.

Finally, the causes that were perceived as feasible targets for process improvement were analysed (Article IV). During the classification of the causes into the process areas and types dimensions, the author of this thesis marked whether the cause was proposed and/or selected as a target for process improvement activities. The causal analysis workshop revealed *detected causes*, whereas the causes the company experts proposed after the workshop are called *proposed causes*. The causes that were selected for process improvement activities by the senior managers are called *selected causes*. The perceived feasibility for process improvement was divided into three importance categories. The selected causes represent the highest-importance category, because such causes reflect the decision makers' perspective. The second-highest importance category is related to the proposed causes because they reflect the company expert's perspective. The third importance category consists of the detected causes, which were neither proposed nor selected for process improvement activities. It was compared quantitatively how the causes in these three importance categories varied. First, the distributions for process areas and cause types were compared. Second, the share of bridge causes was compared with the share of other detected causes.

4. The ARCA method

This section starts by presenting the results from the literature review, including the high-level synthesis of RCA methods and their common steps with work practices. Thereafter, the ARCA method is introduced. These results are presented in Article I, and they are used to answer the first research question (see Figure 2).

4.1 Synthesis of RCA methods from literature

Table 4 summarizes the prior RCA methods and compares them with the ARCA method. There are three steps that are common for RCA methods introduced in the literature. These include target problem detection, root cause detection, and corrective action innovation. These steps and their alternative work practices are discussed below.

Table 4. Summary of RCA methods and their work practices.

Method	Target problem detection	Root cause detection	Corrective action innovation
	Work practices	*Work practices*	*Work practices*
Rooney and Vanden Heuvel (2004)	*Interviewing and inspections*	*Sequence diagram and Decision diagram*	-
Ammerman (1998)	*Paper-and-pencil, walk-through, and flowcharting*	*Sequence diagrams, Interviewing, event and causal factor charts, lists, and worksheets*	*Interviewing*
Latino and Latino (2006)	*Problem sampling, flowcharting sequence diagrams, interviewing, and Pareto analysis*	*Flow chart, logic tree, and meetings with brainstorming*	*Writing individually and meetings*
Card (1998)	*Problem sampling, classification schemes, Pareto analysis, and meetings*	*A fishbone diagram, cause categories, and meetings*	*Meetings*
Dingsøyr et al. (2001)	*Brainstorming, Brainwriting, Post-it notes, and grouping of experiences*	*Selection of the main issues, brainstorming, discussions, a fishbone diagram and drawing causes on a whiteboard*	-
ARCA method (Article I)	*A focus group meeting and brainstorming*	*Anonymous email inquiry, a directed graph, brainwriting and brainstorming in a meeting*	*Email inquiry, brainwriting combined with sceptical and optimistic perspectives, and brainstorming in a meeting*

4.1.1 Target problem detection

RCA methods start with the detection of a target problem. This initial step is usually conducted through problem sampling (Latino and Latino 2006; Andersen and Fagerhaug 2006; Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Grady 1996; Kalinowski, Travassos, and Card 2008; Burnstein 2003), flowcharting (Latino and Latino 2006; Andersen and Fagerhaug 2006; Ammerman 1998), interviewing (Latino and Latino 2006; Rooney and Vanden Heuvel 2004; Rooney and Vanden Hauvel 2003), or brainstorming (Latino and Latino 2006; Andersen and Fagerhaug 2006; Bjørnson, Wang, and Arisholm 2009). Usually, there is a meeting where the target problem is finally decided upon (Card 1998; Burnstein 2003).

Brainstorming in a focus group meeting was included in the work practices of the ARCA method (see Table 4). In the context of software project retrospectives, brainstorming is probably the most cost-efficient approach to detecting the target problems. It has been presented as an excellent approach to identify rapidly what is important to people (Lethbridge, Elliott Sim, and Singer 2005). It has also been presented as a part of lightweight RCA methods (Dingsøyr, Moe, and Nytrø 2001). Additionally, it can be easily conducted in collocated and distributed settings.

Problem sampling, flowcharting, and interviewing were excluded from the ARCA method. Problem sampling sounds like a great idea (see Article I), but it can be used only with problems being reported (Card 1998; Kalinowski, Travassos, and Card 2008; Burnstein 2003; Gursimran and Jeffrey 2009). There are many problems in software projects important to control, but they are not reported, e.g. requirements faults (Gursimran and Jeffrey 2009). Furthermore, flowcharting (Ammerman 1998) might be a useful work practice for the target problem detection, but in the context of software engineering, problems are often intangible. Therefore, drawing a flowchart for the "entire event", in order to explain how the target problem evolves, might be difficult. Interviewing (Latino and Latino 2006; Rooney and Vanden Heuvel 2004) solves the problems of problem sampling and flowcharting. On the other hand, it is a labour-intensive task to meet numerous people, and thereafter register, transcribe, and interpret their answers.

4.1.2 Root cause detection

Root cause detection is the second step of the RCA methods. The outcome of this step is a documented in-depth analysis of the underlying causes of the target problem. Usually, there is a team of people who "investigate" the target problem causes together (e.g. Latino and Latino 2006; Card 1998; Bjørnson, Wang, and Arisholm 2009). The work practices include interviewing (Ammerman 1998), questionnaires (Andersen and Fagerhaug 2006; Burr and Owen 1996), brainstorming, and brainwriting (Latino and Latino 2006; Andersen and Fagerhaug 2006; Burr and Owen 1996). These techniques help to address the target problem causes that many people value highly, which is important. However, as a weakness, none of these approaches fully protects the anonymi-

ty of people. Therefore, it could happen that root cause detection is perceived as “witch hunting” (Latino and Latino 2006).

Furthermore, the detection of the target problem causes usually includes the creation of a cause-effect diagram (see Section 2.3.1). Various diagramming techniques have been introduced, and they can be divided into two subcategories including the list- and network-based structures. List-based structures include a fishbone diagram (Andersen and Fagerhaug 2006; Bjørnson, Wang, and Arisholm 2009; Stålhane 2004; Burnstein 2003; Stevenson 2005), a fault-tree diagram (Andersen and Fagerhaug 2006), a logic tree (Latino and Latino 2006), and a causal-factor chart (Rooney and Vanden Heuvel 2004). Network-based structures include a directed graph (Bjørnson, Wang, and Arisholm 2009) and a matrix diagram (Andersen and Fagerhaug 2006). Furthermore, simple cause lists and worksheets can also be used to organize the target problem causes (Ammerman 1998).

Brainwriting followed by brainstorming in a meeting was included in the work practices of the ARCA method (see Table 4). Brainwriting provides an efficient way to make good use of all participants simultaneously. Instead, brainstorming helps to refine the findings of individuals into more concrete conclusions about the root causes. The prior literature (Kavadias and Sommer 2009) indicates that brainstorming attains better solutions when it is used with cross-functional problems, and brainwriting is better when it is used with complex problems. The problems of software projects are both complex and cross-functional (see Article I). Therefore, using these techniques together is reasonable. Furthermore, the use of the directed graph was also included in the work practices of the ARCA method. The directed graph solves the problem of duplicating cause statements (see Article I). Additionally, the use of the directed graph has been claimed as an effective technique for software project retrospectives (Bjørnson, Wang, and Arisholm 2009).

Interviewing (Ammerman 1998) and questionnaires (Andersen and Fagerhaug 2006; Burr and Owen 1996) were excluded from the ARCA method. Interviewing would have required more effort than keeping a meeting, and the use of questionnaires would have steered the thinking of retrospective participants into some premade topics, potentially biasing the results. Furthermore, these techniques have not been recommended in the prior RCA methods of lightweight software project retrospectives.

4.1.3 Corrective action innovation

Corrective action innovation is the final step of the RCA methods. The outcome includes corrective actions that are developed for the selected target problem causes. The selection of causes should emphasize the level of controllability. The prior literature included very little practical guidance while considering “how to develop corrective actions”. Corrective actions are usually developed in a meeting with a group of people (Andersen and Fagerhaug 2006; Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Grady 1996). Additionally, the use of brainstorming and brainwriting (Andersen and Fagerhaug 2006) are recommended. Interviewing has also been introduced as

an approach to develop corrective actions (Ammerman 1998). However, considering the differences between keeping a meeting and conducting separate interviews, the meeting probably increases the commitment of participants more than the separate interviews.

Furthermore, problem-prevention frameworks (Andersen and Fagerhaug 2006) have been developed to view the solution space of problems from various perspectives. The frameworks include Systematic Inventive Thinking, the Theory of Inventive Problem Prevention, and the Six Thinking Hats (Andersen and Fagerhaug 2006). However, the frameworks are rather difficult to use, and more creative techniques should be used instead (Andersen and Fagerhaug 2006).

Similarly to the step of root cause detection (see Section 4.1.2), the use of brainwriting combined with brainstorming was concluded as the most optimal work practice for the ARCA method. We also found it to be important to take into account the potential "positive" and "negative" effects of the developed corrective actions (Andersen and Fagerhaug 2006).

4.2 Overview of the ARCA method

Figure 6 summarizes the ARCA method (see Article I). The method follows the common steps of prior RCA methods, and its work practices are based on analytical argumentation about the prior methods, discussed in Section 4.1. These steps and their work practices are summarized in the following sub-sections.

4.2.1 Step 1: Target problem detection

The outcome of the first step of the ARCA method is a target problem and a list of named experts who are invited to an in-depth analysis of the target problem. This step includes a focus group meeting lasting approximately 60 minutes. In the meeting, the following issues should be brainstormed, justified, and documented: what is the target problem and why exactly is this problem important to prevent?

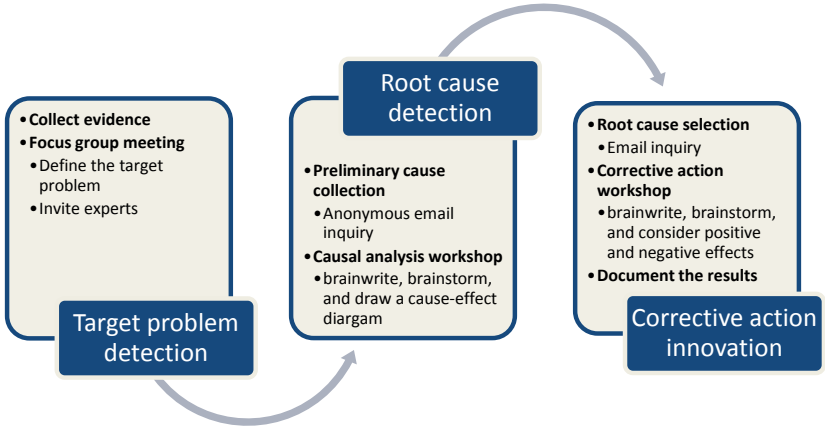

Figure 6. The overview of the ARCA method.

The experts who should analyse the target problem are also considered and selected in the meeting (four to ten experts). When selecting the experts, it is important to consider all relevant stakeholders of the target problem. For example, in the case of software project failures, these may include sales personnel, product managers, project managers, software developers, software testers, and software quality assurance staff.

4.2.2 Step 2: Root cause detection

This is the second step of the ARCA method. After this step, the most important target problem causes are detected and evaluated. Anonymous and public approaches are both important in the detection of target problem causes. This step consists of two work phases: preliminary cause collection and causal analysis workshop.

In preliminary cause collection, the facilitator of the ARCA method sends out an email inquiry to the selected experts in order to collect the target problem causes. The email inquiry should be confidential in order to create a trustworthy knowledge-sharing between the experts and facilitator. The email inquiry forces the experts to consider the target problem in advance. The inquiry asks the experts to list at least five causes of the target problem. Thereafter, the target problem causes are organized into a cause-effect diagram by the facilitator, as presented in Figure 7. Using a software tool is recommended here.

Causal analysis workshop is a meeting wherein the target problem causes are analysed in-depth. The meeting is prepared by the facilitator. A cause entity is defined as “a cause and its sub-causes, which together form an entity that is reasonable to process together” (Article I). By using the cause-effect diagram created in preliminary cause collection, the facilitator selects the cause entities being processed in the meeting. It should be noted that the cause entities could overlap. Processing a cause entity containing approximately ten causes takes about 40 minutes.

In causal analysis workshop, the selected cause entities are extended by detecting new causes. The facilitator starts the meeting by presenting the target problem and its preliminary causes, including the selected cause entities for the experts. The meeting continues by collecting new causes for each selected cause entity one at a time. Each cause either deepens or widens a cause entity. The causes are collected in the following three phases:

1. The experts write down (brainwriting) causes on paper for five minutes (the cause-effect diagram is projected onto the wall)
2. Each expert introduces the causes and explains where they should be registered in the cause-effect diagram
3. The experts briefly discuss the target problem causes and try to brainstorm more causes and causal relationships

After all cause entities have been processed, the experts analyse the cause-effect diagram as a whole. The facilitator leads the experts to identify essential target problem causes and to discuss their level of controllability and impact for the target problem.

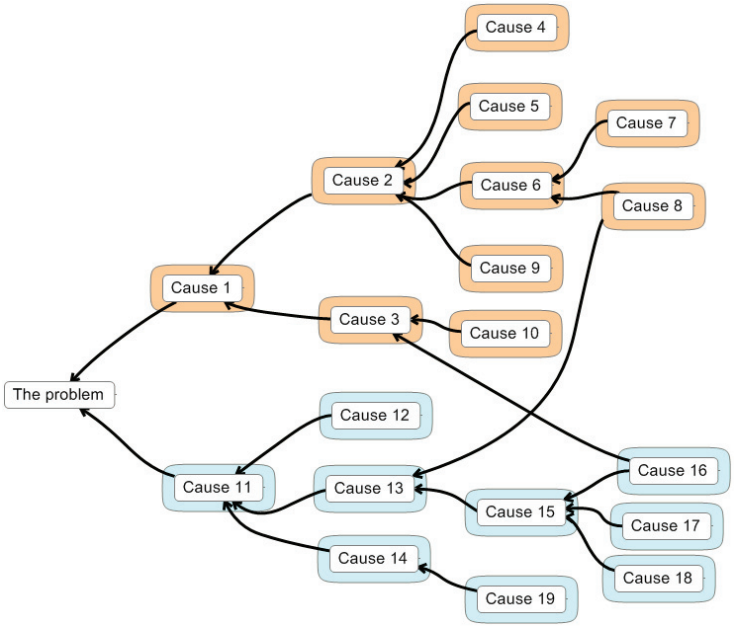

Figure 7. The cause-effect diagram of the ARCA method (Article I).

4.2.3 Step 3: Corrective action innovation

This is the third step of the ARCA method. The outcome of this step is corrective actions addressing the most important target problem causes. The step includes two work phases: root cause selection and corrective action workshop.

Root cause selection is the first work phase of corrective action innovation, and it aims to focus on the development of corrective actions into most feasible targets. The facilitator selects the target problem causes which are processed later in the corrective action workshop. First, the cause-effect diagram is sent to the experts. They are asked to propose target problem causes for which corrective actions should be developed. Additionally, they are asked to evaluate the level of impact on the target problem and the level of difficulty of developing corrective actions for each proposed cause. Thereafter, the facilitator uses his judgment combined with an analysis of the experts' proposals in order to select four to six target problem causes for which the corrective actions will be developed. Each selected cause, including its sub-causes, is documented on an individual paper.

Corrective action workshop is a meeting wherein corrective actions are developed, evaluated, and analysed. The meeting is initialized by the facilitator, who selects the participants to join the meeting. Ideally, the number of participants equals with the number of selected causes (four to six). Furthermore, the participants should be an aggregate of experts being as competent as possible at solving the selected causes. In a corrective action workshop lasting approximately 120 minutes, the selected causes are rotated through the participants. Each participant contributes, in turns lasting ten to fifteen minutes, to one se-

lected cause. The participants develop corrective actions by writing them down on paper. Additionally, they supplement and comment on the corrective actions introduced by other participants. Furthermore, after the corrective actions are developed, they are evaluated. Two attributes are used in the evaluation (scale 1-5): 1) impact on the target problem and 2) feasibility to implement. For each selected cause, the last participant evaluating the corrective actions calculates the sum of evaluations for each corrective action. Thereafter, the participants brainstorm improvements to the corrective action(s) that has/have the highest values in the levels of impact and feasibility.

4.2.4 Step 4: Documentation of the results

The documentation of the results is the final step of the ARCA method. Such a step is not included in every prior RCA method, but in the ones introduced by Card (1998), Dingsøyr et al. (2001), Latino and Latino (2006), and Ammerman (1998).

The final report should cover the target problem definition, including the related background information. This is important in order to communicate the aim of the analysis, including its limitations. The report should also contain the main parts of the cause-effect diagram finalized in the causal analysis workshop. Additionally, the report presents the corrective actions and their evaluations.

The final report is an important source of information, as it can be used to justify why some specific process changes are needed and what corrective actions are relevant to consider. Additionally, the report could improve future analyses, where it could exacerbate preliminary cause collection and help to consider the cause entities for causal analysis workshop. The report could also help to consider the impact and feasibility of corrective actions. Finally, if shared among the company employees, the report could improve organizational learning.

5. ARCA-tool

Software tools are commonly used in RCA to improve the in-depth analysis of problems. However, there are no prior studies on how they support conducting RCA in collocated and distributed software project retrospectives. This section starts with the results of systematic literature review comparison of prior RCA software tools. Thereafter, ARCA-tool, fulfilling the main weaknesses of the prior tools, is presented. These results are introduced in Article III and they are used to answer the second research question (see Figure 2).

5.1 Comparison of RCA software tools

A systematic literature review of 35 prior RCA software tools was conducted in order to evaluate their feasibility for software project retrospectives. The evaluation considered the seven aspects important for conducting a computer facilitated RCA in synchronous software project retrospective meetings (Article III). These aspects are introduced in the following sub-sections. To conclude, the main weaknesses of the prior tools include: 1) lack of real-time collaboration, 2) lack of network structured cause-effect diagrams, and 3) lack of features for voting the RCA method outcome.

5.1.1 Ease of adoption

The first aspect for comparison is the ease of adoption. Software teams rarely have time for retrospectives (Glass 2002), and therefore this is an important aspect. Web browser-based software tools outperform native client software in the ease of adoption. Web browser-based software tools do not require client installation and they can be used from various physical locations with different computers having different operating systems and hardware. Only four existing tools are web browser-based software.

5.1.2 Real-time collaboration

The second aspect is the support for real-time collaboration. Global software engineering is an increasing trend in today's software business (Herbsleb and Moitra 2001), but through the distributed team members, it creates a major challenge for retrospectives. The team members cannot meet face-to-face. Thus, the RCA software tool has to support real-time collaboration over distributed sites in order to make it possible for the participants to contribute to the analysis as it takes place. Obviously this requires that the outcome of the

tool stays in sync between the distributed sites. There are only six tools that fully support real-time collaboration.

5.1.3 Cause-effect diagramming

The third aspect is the support for cause-effect diagramming. The core component of RCA is the analysis of the underlying causal structures of the target problem. In retrospectives, such an analysis is usually conducted by using a cause-effect diagram (Bjørnson, Wang, and Arisholm 2009; Dingsøyr 2005). The majority of the existing tools enable the creation of a cause-effect diagram. However, most of them support only tree-structured diagrams, whereas only three existing tools support the creation of network-structured diagrams.

5.1.4 Corrective action development

The fourth aspect is the support for developing corrective actions for the causes of problems. The software tool should enable developing and linking the corrective actions to the related target problem causes. It seems that the majority of the existing tools fulfil this aspect.

5.1.5 Support for voting

The fifth aspect is about the team commitment through voting. In order to focus the steps of root cause detection and corrective action innovation to the findings that the experts value the most, the software tool should support voting. This way the experts can focus their attention on the causes perceived as the most important. Respectively, they can collaboratively decide the corrective actions that should be implemented. This aspect is supported only in one of the existing tools.

5.1.6 Support for knowledge management

The sixth aspect is the support for knowledge management. It has been claimed that retrospectives can be used to leverage knowledge from individuals to organizations (Dingsøyr 2005). Additionally, it has been claimed that an organizational learning system includes a “global knowledge base” that combines the knowledge (Lee, Courtney, and O'Keefe 1992). Thus, the RCA software tool should include a knowledge base and enable combining the findings of many retrospectives. Such an aspect is supported by the majority of the existing tools.

5.1.7 Costs

The seventh aspect considers the costs of the tools. Only three of the existing tools are free to use, whereas most of the tools are subject to a fee. Thus, there are only a few open-source alternatives available.

5.2 Overview of ARCA-tool

ARCA-tool is a browser-based software that uses a client-server architecture with push-and-pull technology. It solves the main weaknesses of prior RCA tools for software project retrospectives. The software supports distributed real-time collaboration including features for 1) collaborative cause-effect diagramming and 2) the development of embedded corrective actions to the causes of the target problem. The tool also supports knowledge management and organizational learning by enabling capturing, analysing, summarizing, and managing the outcome of one-to-many retrospectives.

ARCA-tool was designed to be used in the retrospectives of software projects with the ARCA method. Additionally, the tool was required to fulfil the seven aspects important for conducting RCA in software project retrospectives, introduced in Section 5.1. This section presents how to use ARCA-tool with the ARCA method.

5.2.1 Initializing ARCA-tool

In order to conduct the ARCA method, the facilitator initializes ARCA-tool by creating an RCA case, which is thereafter shared with the participants of the steps of target problem detection, root cause detection, and corrective action innovation. The participants join the case from their own computers through a TCP network connection. The process support for the different steps of the ARCA method is introduced in the following sub-sections.

Figure 8 summarizes the key features of ARCA-tool embedded in a radial menu, which is activated when a user selects a cause in the diagram. The key features include: Thumb-up (=vote for this cause), Pencil (=edit this cause), Trashcan (=delete this cause), Light bulb (=create a corrective action), Arrow left (=link this cause to another existing cause), + sign (=create a cause that is linked to this cause), Ticket (=classify this cause). More details of the tool can be found in Article III.

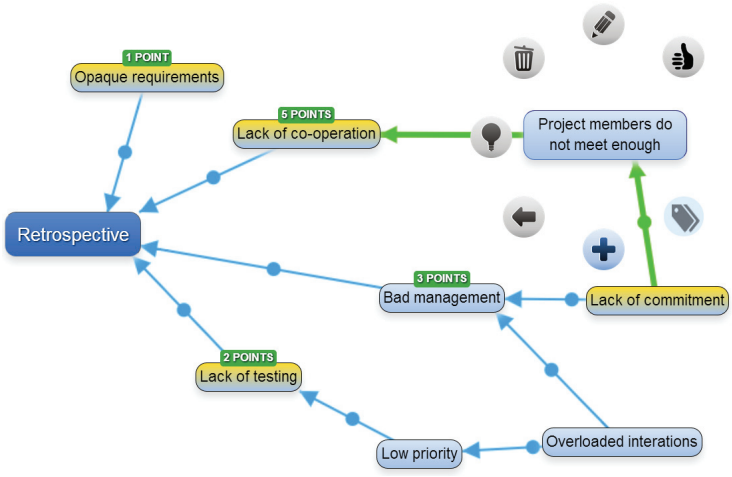

Figure 8. Screen view of ARCA-tool (Article III).

5.2.2 Target problem detection

ARCA-tool supports the step of target problem detection (see Section 4.2.1), which includes a focus group meeting. ARCA-tool can be used in the focus group to register the target problem and the motivation to prevent it. This documentation can be included in the cause-effect diagram of the case.

The diagram can be shared with all relevant stakeholders representing the experts invited to the case. The tool sends the invitations automatically for the defined experts. Alternatively, the invitations can be shared manually by sending the case URL address.

5.2.3 Root cause detection

ARCA-tool supports the step of root cause detection which includes the work phases of preliminary cause collection and causal analysis workshop (see Section 4.2.2). In the phase of preliminary cause collection, the tool allows the defined experts to contribute to the cause-effect diagram of the case "before" the causal analysis workshop meeting. Thus, the facilitator does not need to send a "confidential email" to experts and thereafter organize the causes replied in the email, as the causes are already organized to the diagram by the experts. Additionally, the tool protects the anonymity of the experts.

Furthermore, ARCA-tool supports conducting causal analysis workshop with its features for distributed and collocated knowledge sharing. First, the detected causes can be written down to a cause-effect diagram by the facilitator, acting as a scribe. The diagram can be simultaneously projected on the wall in order to visualize the analysis outcome to the participants. Second, each participant can also contribute directly to the cause-effect diagram from their own computers. There the detected causes are immediately visible for other participants. Thus, they can also contribute to the findings of others in real-time. The workload of the facilitator also decreases since there is no need for a scribe during the meeting. Third, if combined with an online audio bridge, ARCA-tool enables conducting the ARCA method as distributed. The geographically distributed experts can register, introduce, and discuss the target problem causes similarly than with collocated settings.

5.2.4 Corrective action innovation

ARCA-tool enables the conducting of corrective action innovation (see Section 4.2.3), which includes the work phases of root cause selection and corrective action workshop. In ARCA-tool, the participants can vote the causes they perceive as important by "liking" the causes (see "Points" in Figure 8). The amount of "likes" for each cause is limited to +/-1 for the experts while remaining unlimited for the facilitator. This feature makes it possible for the facilitator to ask the experts to propose the causes they perceive as important to proceed further in the process improvement activities. Respectively, the facilitator can emphasize the causes that are selected to process improvement activities by using the voting feature.

According to the ARCA method description, the corrective actions are developed by "writing them down on paper and rotating them through the participants" (Article I). In ARCA-tool, the corrective actions are developed by embedding them to the related causes. The tool does not include features in allocating the selected causes for one expert only. Instead, the tool makes it possible to contribute to any of the causes registered to the cause-effect diagram. The other participants cannot modify or comment on the corrective actions developed by others, but they can register a new corrective action refining the existing ones. Finally, the participants can vote corrective actions by using the liking feature of the tool. Instead, there are no specific forms for "feasibility" and "impact" evaluations.

5.2.5 The documentation of results

The ARCA method ends with the documentation of results (see Section 4.2.4), which is also supported by ARCA-tool. While the facilitator combines the gained knowledge from the case, the tool makes it possible to save the outcome of the RCA case as a *.CSV file, which includes the causes, corrective actions, and their related votes.

Additionally, ARCA-tool enables conducting further analyses of the retrospective outcome including the analysis of the cause types, process areas, and causal relationships of problems. These features promote consideration for what the problem causes are, where they occur, and why they occur (Article IV). Each cause can be classified into the *type* and *process area* dimensions. The user can use the default dimensions, introduced in Article IV, or develop their very own dimensions that are more feasible for their context of use. Thereafter, the tool provides statistics about the distributions of causes, regarding their status in the RCA case (*detected causes*, *proposed causes*, *causes with elimination ideas*), in both of these dimensions simultaneously as a table view, or separately as a pie chart view. Furthermore, the tool can draw a graph summarizing the relationships over the process areas. The tool can also be used to view the internal and external causes for a process area. These analyses can be included in the final report of the ARCA method.

ARCA-tool supports organizational learning and knowledge management by providing features for monitoring the outcome of many RCA cases. As a limitation, the users can analyse only the outcome of the cases they have participated in. The tool can be used to manage, monitor, and analyse the outcome of an individual RCA case as well as the combination of many cases. The status of the detected causes (*detected, elimination, won't fix, fixed*) and corrective actions (*idea, will be implemented, implemented, rejected*) can be managed. Furthermore, the users can filter the outcome they are interested in to monitor (causes, corrective actions, and specific RCA cases).

6. Evaluation results

This section presents the evaluation results from the field studies and student experiment. These are summarized in Table 5 and introduced in detail in Sections 6.1 and 6.2. They answer the research questions RQ3 and RQ4. Furthermore, Section 6.3 presents the results of the multiple case study, which answers the research questions RQ5-RQ7.

6.1 Evaluation of the ARCA method

In comparison to the existing practices of the industrial cases, the ARCA method was perceived as efficient. Respectively, the method was perceived as easy to use. The detected causes were also perceived as accurate and they helped to develop high-quality corrective actions.

The ARCA method was evaluated from different perspectives. Cases 1-4 (N=30) evaluated all steps and work practices of the method. Cases 5 (N=11) - 6 (N=5) and student experiment (N=51) evaluated only the step of root cause detection (ARCA ltd.). Cases 5-6 also evaluated the support of ARCA-tool for the ARCA method. Furthermore, the student experiment included the comparison of the number of detected causes between the ARCA method and control group.

Table 5. Summary of evaluation results regarding the ARCA method and ARCA-tool.

Evaluation	**Field Studies (N=46)**						**Experiment (N=51)**	
Case	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	ARCA	Control
Method	ARCA	ARCA	ARCA	ARCA	ARCA *	ARCA *	ARCA *	ARCA *
Root cause detection step								
Ease of use	Mod	Mod	High	Mod	High	High	High	High
Efficiency	High	High	High	Mod	High	High	Increase#	Decrease#
Accuracy	High	High	High	High	High	High	-	-
Corrective action innovation step								
Ease of use	High	High	High	High	-	-	-	-
Efficiency	High	High	High	High	-	-	-	-
ARCA- tool								
Usefulness	-	-	-	-	High	High	-	-
Ease of use	-	-	-	-	High	High	-	-

Interpretation scale (based on questionnaires): Low (avg. < 4), Mod (avg. >4&<5), High (avg. > 5), * The limited ARCA method (ARCA ltd.), # Based on the number of detected causes

6.1.1 Evaluation of the ARCA method ease of use

Table 6 summarizes the results from the questionnaires regarding the ease of use of the ARCA method. The interviews consolidate these results. We can see from the table that the ARCA method was perceived as generally easy to use. On the other hand, comparison between the cases reveals that the perceived ease of use increased in Cases 5-6 and the student experiment in contrast to Cases 1-4.

In Cases 1-4, the participants evaluated in the questionnaires that the ease of use of the corrective action innovation step is "high" (avg. > 5) and the root cause detection step is "moderate" (4< avg. <5). Respectively, in the case interviews (see Article I), the ARCA method was generally experienced as easy to use. This was a common opinion over the key representatives of each case. On the other hand, organizing the detected causes was noted to be a challenging task. It was also said that the assistance of the researchers made the ARCA method unnaturally easy to use.

In Cases 5-6, the participants evaluated in the questionnaires that the ease of use of the root cause detection step is "high". They evaluated the "easiness to collect causes" with high values and the "easiness to detect root causes" with moderate to high values (see Table 6). Respectively, the interviews indicated that the participants perceived the method as simple and intuitive (see Article III). It was also noted that the perceived difficulty of the analysis is dependent on the number of the detected causes. Furthermore, in Case 6, the participants evaluated that the ease of use of the ARCA method makes an improvement over their existing practices (see Article III).

In the student experiment, similar results regarding the ease of use were found. Regarding the results from questionnaires, the students perceived that the ease of use of collecting the target problem causes is "high" (see Table 6), but they also perceived that the difficulty of detecting the problem causes is

Table 6. Evaluation results regarding the ease of use of the ARCA method.

Ease of use	**Field Studies (N=46)**						**Experiment (N=51)**	
Case	Case 1	Case 2	Case 3	Case 4	Case 5*	Case 6*	ARCA	Control
Root cause detection step								
Cause collection	-	-	-	-	High^	High	High	High
avg.					5.9	5.6	5.6	5.6
std.					0.8	1.0	1.1	1.0
Cause detection	Mod	Mod	High	Mod	High^	Mod	-	-
avg.	4.3	4.9	5.1	4.8	5.6	4.2		
std.	0.8	1.2	1.2	0.4	0.9	1.0		
Corrective action innovation step								
Dev. method	High	High	High	High	-	-	-	-
avg.	5.7	6.0	6.0	6.0				
std.	1.0	0.8	0.6	0.6				

Scale: 1=very low, 4=moderate, 7=very high, * normalized scale 1-7 (the original scale was 1-5), ^combined results from three teams

"high" (see Article V). Furthermore, the students compared the cause-effect diagram of the ARCA method (see Section 3.6.2) with the list-of-causes technique (control group). Considering the results from the comparison, most of the students evaluated that the cause-effect diagram of the ARCA method is a "good" technique to organize the causes (Median=6). Instead, they evaluated that the list-of-causes technique is only "somewhat good" (Median=5). The difference between these two techniques is statistically significant (p=0.001).

6.1.2 Evaluation of the ARCA method cost-efficiency

Table 7 summarizes the results from the questionnaires regarding the perceived cost-efficiency of the ARCA method. We can see from the table that the ARCA method was perceived as cost-efficient at each case (avg. of usefulness & efficiency are both >4). The interviews consolidate these results.

In Cases 1-4, the results from questionnaires indicate that the steps of root cause detection and corrective action innovation are both useful and efficient. Respectively, the results from interviews indicate that the ARCA method was perceived as cost-efficient (see Article I). The interviews revealed that the key representatives experienced that their companies should adopt the ARCA method. They also perceived that the case results were beneficial in contrast to the effort used. Additionally, they were not able to name any other method that could reach equally advantageous results with lower costs.

In Cases 5-6, the limited ARCA method was perceived as cost-efficient. In Case 5, the limited ARCA method was used previously, which indicates that the method was already found to be feasible and applicable to software project retrospectives. Respectively, in Case 6, wherein the limited ARCA method was compared with the existing practices, the participants evaluated in the questionnaires that the cost-efficiency of the method is high. Furthermore, regarding the results from interviews, the participants from both cases experienced that the structured approach of the limited ARCA method is one of its advantages. They also experienced that the method helps to detect the causes of problems.

In the student experiment, the participants evaluated that the limited ARCA method is useful. They also evaluated that the cost-efficiency of the method is high. Additionally, the statistical analyses on the method outcome indicate that the limited ARCA method slightly increased the method effectiveness (p=0.065, Cohen's d=0.57). It also increased the number of hub causes (p=0.010, Cohen's d=1.42). Furthermore, the group interviews with students revealed concepts supporting the use of the cause-effect diagram in the ARCA method. The students perceived that in contrast to the control group, the cause-effect diagram of the limited ARCA method helped to outline how the causes are related to one another. Respectively, the visual structure of the cause-effect diagram was perceived as feasible for RCA. The cause-effect diagram was also perceived as a visually easier technique to navigate the detected causes. The students claimed that the cause-effect diagram helped to focus and process the detected causes systematically. The only argument that supported the control group is the high readability of the list-of-causes technique.

Table 7. Evaluation results regarding the cost-efficiency of the ARCA method.

Cost-efficiency	**Field Studies (N=46)**						**Experiment (N=51)**	
Case	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6*	ARCA	Control
Root cause detection step								
Usefulness	High	High	High	High	-	-	High	High
avg.	5.4	5.8	5.8	5.3			6.2	6.2
std.	0.4	0.6	0.5	0.6			0.8	0.8
Efficiency	High	High	High	Mod	-	High	High	High
avg.	5.4	5.1	5.3	4.8		5.9	6.0	5.8
std.	1.2	1.0	1.2	1.3		0.6	1.0	1.0
Effectiveness#	-	-	-	-	-	-	Increase	Decrease
avg.							107	94
std.							22	22
Corrective action innovation step								
Usefulness	Mod	High	High	High	-	-	-	-
avg.	4.8	5.0	5.1	5.0				
std.	1.0	1.1	1.1	0.6				
Efficiency	High	High	High	High	-	-	-	-
avg.	6.2	6.0	6.1	6.3				
std.	1.0	0.9	0.7	0.5				

Scale: 1=very low, 4=moderate, 7=very high, * normalized scale 1-7 (the original scale was 1-5), # Effectiveness indicates the number of detected causes

6.1.3 Evaluation of the ARCA method outcome

The outcome of the ARCA method includes the causes of the target problem and the related corrective actions. Table 8 summarizes the results from the questionnaires regarding the perceived correctness of the detected causes and the perceived impact and feasibility of the developed corrective actions. The results from the interviews consolidate these results.

Regarding the correctness of detect causes, the participants of Cases 1-4 perceived that the detected target problem causes were correct (see Table 8). Additionally, the participants evaluated that feasible corrective actions that have a high impact on the target problem were developed. Respectively, the interviews with the key representatives indicate that "significant root causes" were detected with respect to the target problems (see Article I). Most of the key representatives also believed that, if implemented, the corrective actions would have a high impact on the prevention of the target problem. As an exception, it was claimed in Case 2 that the corrective actions don't prevent the target problem, but they assist the company to make improvements in their processes. Furthermore, in Cases 5-6, the participants evaluated the correctness and impact of the detected causes with high values. Respectively, regarding the results from interviews, they perceived that correct target problem causes were detected (see Article III).

Figure 9 presents a scatter chart of the developed corrective actions in Cases 1-4. A high quality corrective action is "highly feasible and equally effective" (Article I). In Cases 1-4, each corrective action of each case was evaluated by

the case participants. The evaluations were conducted by using a symmetric ordinal scale from one to five (1=low, 2, 3, 4, 5=high). We can see from Figure 9 that the share of high-impact (avg. >= 3) corrective actions was larger than the share of low-impact (avg. < 3) corrective actions in each case. Instead, the share of high-feasibility (avg. >= 3) corrective actions was larger than the share of low-feasibility (avg. < 3) corrective actions in Cases 1 and 4 only. It is probably easier to develop high-impact corrective actions than to make them feasible. Despite the difficulties to develop high-quality corrective actions (avg. impact & feasibility are both >= 3), such corrective actions were developed in each case, as can be seen from the figure.

Table 8. Evaluation results regarding the outcome of the ARCA method.

Outcome evaluation	**Field Studies (N=46)**					
Case	Case 1	Case 2	Case 3	Case 4	Case 5*	Case 6*
Root cause detection						
Correctness of causes	High	High	High	High	High^	High
avg.	6.0	5.8	6.2	5.5	5.7	5.3
std.	0.5	0.7	0.8	0.8	0.8	0.6
Impact of causes	-	-	-	-	High^	High
avg.					5.5	5.3
std.					1.3	1.2
Corrective action innovation						
Impact of ideas	High	High	High	High	-	-
avg.	5.6	5.4	5.9	5.3		
std.	0.5	0.7	0.7	0.8		
Feasibility of ideas	High	Mod	High	High	-	-
avg.	5.3	4.4	5.3	5.7		
std.	0.5	1.1	0.8	0.8		

Scale: 1=very low, 4=moderate, 7=very high, * normalized scale 1-7 (the original scale was 1-5), ^combined results from three teams

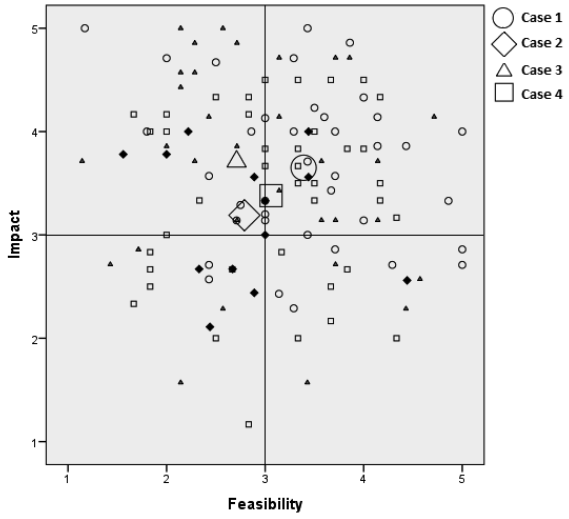

Figure 9. Scatter chart of the perceived impact and feasibility of individual corrective actions (small marks) and their averages in the cases (large marks).

6.2 Evaluation of ARCA-tool

This section presents the empirical results regarding the evaluations of ARCA-tool. The tool was evaluated in Cases 5-6 only (article III). Table 9 summarizes the results from the questionnaires regarding the perceived ease of use and usefulness of ARCA-tool. Our results indicate that ARCA-tool increases the cost-efficiency of the limited ARCA method. Additionally, the tool is perceived as essential when the ARCA method is conducted with geographically distributed settings.

6.2.1 Evaluation of the ease of use of ARCA-tool

ARCA-tool was perceived as easy to use and learn in Cases 5-6. In Case 5, the participants evaluated the ease of use and learnability of the tool with very high values (see Table 9). In Case 6, the participants evaluated the ease of use and learnability of the tool with high values. The values given in Case 6, however, were lower than in Case 5. Case 5 had used the tool previously, whereas the tool was new to the participants of Case 6; a difference between the cases could explain the differences in the evaluations.

The results from the interviews consolidate the results from questionnaires (see Article III). In Case 5, the participants experienced that the tool makes it easier to visualize the outcome of the root cause detection step, i.e. the causes of the target problem (the term "retrospective" was used in Article III to refer to this step). In Case 6, the tool was characterized as "intuitive" and "relatively easy to use". Additionally, the interviews at Case 6 indicate that it is positive that only the necessary features are included in the tool. It was also claimed in the interviews of Case 6 that the perceived difficulty of using ARCA-tool in the step of root cause detection correlates with the number of detected causes. The

Table 9. Evaluation results regarding ARCA-tool.

ARCA-tool	**Field studies (N=16)**			
Case	Case 5*			Case 6*
	Team 1 (N=3)	Team 2 (N=5)	Team 3 (N=3)	Team 4 (N=5)
Ease of use	Very high	High	Very High	High
avg.	7.0	6.4	7.0	5.6
std.	0.0	0.7	0.0	1.0
Learnability	High	High	Very High	High
avg.	6.6	6.4	7.0	5.6
std.	0.8	0.7	0.0	1.4
Cost-efficiency	-	-	-	High
avg.				5.6
std.				1.4
Usefulness	High	High	High	High
avg.	6.0	5.9	6.6	5.6
std.	0.8	1.1	0.7	1.0

Scale: 1=very low, 4=moderate, 7=very high, * normalized scale 1-7 (the original scale was 1-5)

interview results from Case 6 also indicate that the tool could be improved. One participant claimed that when the causes of the target problem are organized, the visualization of cause groups would be important and it is currently difficult with the tool.

6.2.2 Evaluation of the usefulness of ARCA-tool

In both cases, the results from the questionnaires indicated that the tool helped to detect the causes of problems. In Case 5, the participants evaluated that the efficiency of the step of root cause detection step would be lower without the tool (see Article III). Respectively, the participants of Case 6 evaluated that, in comparison to their previous practices, the cost efficiency of the tool is high (see Table 9). Additionally, the participants of both cases evaluated that the "assistance of the tool for cause detection" is significant (see Article III).

Regarding the results from the interviews, it seems to be a common opinion that the tool improves the limited ARCA method (see Article III). Additionally, the results indicate that the tool is essential in geographically distributed settings. Furthermore, the interviews of Case 5 indicate that in face-to-face settings, the tool can be substituted with a whiteboard and Post-it notes, an approach introduced by Stålhane et al. (2003). However, the efficiency of analysis would then decrease (see Article III).

6.3 The cause types, process areas, and their relationships

This section summarizes the case study results regarding the outcome of the ARCA method in Cases 1-4 (articles II and IV). The results indicate that in a case of software project failure, the outcome of the ARCA method helps to explain what happened, where, and why.

6.3.1 Process areas

The ARCA method outcome included causes from five process areas. These included "Management", "Sales & Requirements", "Implementation", "Software Testing", and "Release & Deployment". A total of 97.8 % to 100% of the detected causes were related to the process areas at each case.

The process areas of the detected causes are somewhat similar to the ones found in software engineering process literature, e.g. RUP (Jacobson, Booch, and Rumbaugh 1998) and the waterfall model (Royce 1970). This means that most of the detected causes indicated commonly accepted development process areas wherein they occurred.

6.3.2 Cause types

Table 10 summarizes the cause types and their sub-types. We can see from the table that the ARCA method outcome included causes with four types. These included "People", "Tasks", "Methods", and "Environment". The cause types are similar to the ones introduced in the literature of the causes of software

Table 10. Summary of cause types and their sub-types.

Type	Sub-type	Examples
People	Instructions & Experience	*Lack of instructions when and how to verify.*
	Values & Responsibilities	*People do not care if the number of bugs increases.*
	Cooperation	*Miscommunication between the developers and testers.*
	Company Policies	*New issues are not registered.*
Tasks	Task Output	*Requirements are insufficient.*
	Task Difficulty	*It is difficult to create a comprehensive specification.*
	Task Priority	*The priority of defect fixing is too low.*
Methods	Work Practices	*Implementation is done directly in the test environment.*
	Process	*The process for software testing is missing.*
	Monitoring	*An opaque view of the quality during the development work.*
Environment	Existing Product	*The structure of the product has decayed during the past.*
	Resources & Schedules	*Lack of time to report defects specific enough.*
	Tools	*The version control system does not support customization.*
	Customers & Users	*Importance for the customers is not well defined.*

engineering problems (see Article IV). Furthermore, the detected causes were also expressed with more details than these coarse-grained cause types. The sub-categorization of the detected causes revealed a total of fourteen different sub-types divided into three to four sub-types for each cause type. The sub-types are also in line with the findings of prior studies (McLeod and MacDonell 2011).

6.3.3 Similarities of the causes of failures

The causes of project failures were different in terms of process areas, but similar in terms of cause types. The distributions of causes in process areas were case dependent, which means that regarding the process areas of the detected causes, the failures were different. Instead, regarding the cause types, the cases were similar.

In each case, the cause types were equally distributed into People (avg. 29%, std. 6%), Tasks (avg. 26%, std. 4%), Methods (avg. 22%, std. 3%), and Environment (avg. 22%, std. 5%). All of these cause types were also frequent in all process areas. The cases were also similar in terms of seven sub-types, covering 81% of all detected causes on average (std. 2%). These sub-types included Instructions & Experience (avg. 16%, std. 4%), Values & Responsibilities (avg. 8%, std. 6%), Work Practices (avg. 16%, std. 4%), Task Output (avg. 16%, std. 2%), Task Difficulty (avg. 7%, std. 3%), Existing Product (avg. 7%, std. 5%), and Resources & Schedules (avg. 9%, std. 4%).

Considering the bridge causes (see Section 3.7.2), the commonality between the cases was that 1) the bridge causes were frequent in the detected causes (avg. 50%) and 2) the company experts (avg. 56%) and key representatives (avg. 68%) perceived them as feasible targets for process improvement activities. This means that the company people perceived it important to control the

causes of software project failures, which are related to possible causal relationships over the process areas.

6.3.4 Common causal relationships bridging the process areas

Similar causes were related to similar causal relationships. Figure 10 summarizes the common causes of project failures and their related causal relationships bridging the process areas together. The term "common" refers to a cause that occurred in at least three of our four cases. Such a definition is in line with prior studies (e.g. Cerpa and Verner 2009; Verner, Sampson, and Cerpa 2008).

The common causal relationships bridging the process areas included *Weak Task Backlog*, *Lack of Cooperation, and Lack of Software Testing Resources*. Weak Task Backlog bridged the Sales & Requirements, Management, Implementation, and Software Testing process areas (this process area was only relevant in two cases). Lack of Cooperation bridged the Sales & Requirements, Implementation, and Software Testing process areas. Lack of Software Testing Resources bridged the Management and Software Testing process areas. Furthermore, these three common causal relationships were also interconnected to one another.

The common causal relationships alone did not cover any of the cases. When the case specific results were compared, it was found that each failure was also caused by different, case-specific causes that were interconnected to one another differently. Additionally, the common causes were neither proposed nor selected at every case, and other than common causes were proposed at every case. This means that the software project failures could not have been explained by using the common causal relationships alone. The common causal relationships could only improve the knowledge related to the "possible" causes of software project failures.

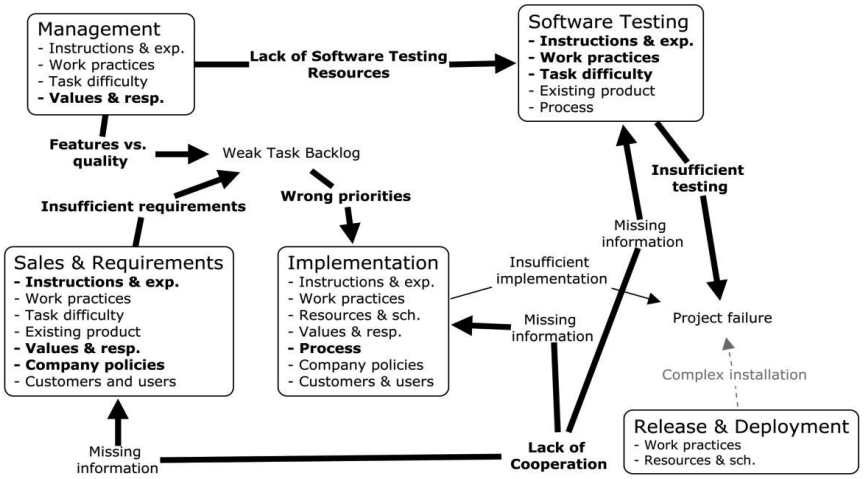

Figure 10. Common causes and bridging causal relationships found in at least three out of four cases (Article IV) (Bolded text/line indicates the selected causes; Normal text/line indicates the sub-causes of the selected causes; Dashed line/grey text indicates that the cause was neither a selected cause nor a sub-cause; Lines with arrows and text between the process areas indicate the direction of causal relationships interconnecting the process areas).

7. Discussion

This section answers the research questions and discusses their main implications. Additionally, the section presents the main threats to validity.

7.1 Lightweight RCA method and software tool

Literature review about the prior RCA methods and the systematic investigation of the environment of use was utilized to create the ARCA method and ARCA-tool. These results contribute to the first research problem: *How can RCA be conducted in collocated and distributed software project retrospectives?* Two research questions were stated for this research problem and they are answered next.

7.1.1 Common steps of RCA methods and their work practices

Section 4.1 summarizes the synthesis of the common steps and work practices of prior RCA methods. Section 4.2 presents the ARCA method. These results answer the first research question discussed below.

RQ1: What are the common steps of RCA methods, and how are they to be conducted?
The concrete work practices of RCA are fairly little studied in the context of software project retrospectives. Therefore, synthetizing the steps of prior RCA methods and their work practices (see Section 4.1) was an important contribution to the prior studies (see Section 2.3.1). Respectively, the ARCA method (see Section 4.2) is an important contribution to the prior literature as it concretizes how to conduct lightweight RCA over the common steps of prior methods including the steps of target problem detection, root cause detection, and corrective action innovation. Additionally, the ARCA method makes a good starting point for the industrial evaluation, as it provides measurable RCA construction to increase the comparability of the evaluation results over the different cases.

The main difference between the prior RCA methods is their different work practices in the step of target problem detection. Problem sampling has been commonly used in large organizations, whereas it requires too much effort in order to be feasible for SME organizations. Instead, brainstorming in a meeting has been introduced as a feasible approach to SME organizations, whereas it has not been recommended for large organizations. Regardless of the organization size, the used work practices should reveal “actual problems” instead of

"subjective opinions" (Bjarnason and Regnell 2012). In SME organizations, it could be more cost-efficient to detect an "actual" target problem by using brainstorming in a meeting than by using problem sampling. Therefore, such an approach was included to the ARCA method. Instead, the situation might be opposite in large organizations. The need for problem sampling, including defect sampling and project surveys, could increase among the increasing number of employees and organizational complexity.

Furthermore, the prior RCA methods are mostly similar in the step of root cause detection, where the causes of the target problem are analysed in-depth. The ARCA method follows the prior methods by using brainstorming and brainwriting in a causal analysis meeting in order to create a cause-effect diagram of the target problem causes (Card 1998; Bjørnson, Wang, and Arisholm 2009). Instead, the ARCA method differs from the prior methods by protecting the anonymity of participants.

Most of the prior RCA methods also include the step of corrective action innovation, which develops action proposals for the most controllable and important root causes. The action proposals are usually developed in a meeting with a group of people (Andersen and Fagerhaug 2006; Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Grady 1996). Brainstorming and brainwriting (Andersen and Fagerhaug 2006) have been presented as useful work practices, which were also included in the ARCA method.

Considering the RCA work practices, it becomes reasonable to claim that the outcome of RCA is only a reflection of expert knowledge instead of true reality. The RCA methods are highly dependent on data investigation techniques including brainstorming, brainwriting, and interviewing. These techniques are often used at every step of RCA methods. Therefore, it is possible that the use of RCA results in wrong conclusions and inaccurate corrective actions.

7.1.2 Software tools for the RCA of retrospectives

Section 5.1 summarizes the comparison results of the prior RCA software tools and Section 5.2 introduces ARCA-tool. Together, these results contribute to the second research question discussed below.

RQ2: What software tools for RCA are introduced, and how do they support software project retrospectives?

Software tools could help to improve the RCA of software project retrospectives. We found that the use of Post-it notes and a whiteboard during RCA (Stålhane et al. 2003; Bjørnson, Wang, and Arisholm 2009) should be substituted with a monitor and software tool (see articles I and III). We also found the challenge of conducting RCA in distributed software project retrospectives due to the lack of real-time collaboration tool support (see Article III). Therefore, developing ARCA-tool was reasonable.

There are at least seven important aspects that should be considered while evaluating the software tool for RCA (see Section 5.1). These aspects can be divided into 1) technical features and 2) features for RCA. Considering the technical features, the RCA tool should support real-time collaboration. Addi-

tionally, the tool should be easy to adopt. Considering the RCA features, the software tool should support cause-effect diagramming, corrective action development, voting, and knowledge management.

A total of 35 software tools for RCA were found by using a systematic literature review. Regarding the comparison results, it seems that the prior RCA tools support software project retrospectives inadequately. The software tools for RCA include mostly proprietary native client software, which are developed for an individual analyst who investigates target problems by using interviewing techniques.

7.2 Perceived ease of use and cost-efficiency

Field studies combined with the student experiment were used to evaluate the ease of use and cost-efficiency of the ARCA method and ARCA-tool. These results contribute to the second research problem: *Is RCA perceived as efficient and easy to use in software project retrospectives?* Two research questions were stated for this research problem and they are answered next.

7.2.1 Ease of use and cost-efficiency of the ARCA method

The ARCA method was evaluated with target problems at different company levels starting from the company-level problems of software project failures and ending with the team-level problems of individual software development teams. Additionally, the method was evaluated in collocated and distributed retrospectives. The evaluation results regarding the ARCA method are summarized in Section 6.1, and they contribute to the third research question discussed below.

RQ3: Is the ARCA method perceived as efficient and easy to use for analysing software engineering problems in software project retrospectives?
Regarding the evaluation results, the ARCA method was perceived as cost-efficient and easy to use. This was the case in collocated and distributed software project retrospectives, which covered analyses between the top-level target problems (Article I) and team-level target problems (articles III and V). In each case, the effort used was perceived as suitable in terms of the output of the method. The detected causes were also experienced as correct in contrast to the target problems. Additionally, high quality corrective actions were developed in each case where the step of corrective action innovation was conducted (Cases 1-4). Furthermore, in each case, the method was perceived as useful.

These evaluation results indicate that RCA is an important part of software project retrospectives, as also indicated in the prior studies on post-mortem reviews (Stålhane et al. 2003; Bjørnson, Wang, and Arisholm 2009; Dingsøyr 2005; Collier, DeMarco, and Fearey 1996) and defect causal analysis (Card 1998; Leszak, Perry, and Stoll 2000; Jalote and Agrawal 2005; Gupta et al. 2008; Grady 1996; Kalinowski, Travassos, and Card 2008; Jacobs et al. 2005; Nakashima et al. 1999). In Cases 1-4 and 6, the existing retrospective practices

did not include RCA, but they did include the detection of problems and development of corrective actions. In the existing practices, the structured investigation of the underlying causes of problems, i.e. RCA, was substituted with "informal discussions about problems" (articles I and III). Respectively, cause-effect diagrams were not used to register the findings of software project retrospectives. When the ARCA method was used in the cases, the participants perceived it as cost-efficient and feasible for their needs.

There are at least three hypotheses as to why the ARCA method improved the existing practices. First, the structured investigation approach of the ARCA method decreased the level of informality by providing "process structure" (Dennis et al. 1997). Second, the use of brainwriting in order to detect the causes of problems (Andersen and Fagerhaug 2006) decreased the problem of dominating team members who speak over the others (Article III). Third, the visual power of the cause-effect diagram decreased *memory bias* (Von Zedtwitz 2002) by helping the participants to remember all relevant findings and outline them as a whole (Eden 2004).

Considering the three hypotheses listed above, the results indicate that the structured investigation approach is the key for successful software project retrospective, which can be additionally improved by using brainwriting and the cause-effect diagram. In the student experiment, the perceived usefulness of the ARCA method was not significantly dependent on the use of the cause-effect diagram. However, the detailed comparison of the method outcome and the analysis of the perceptions of participants revealed that the cause-effect diagram was a more optimal technique for visualizing the outcome of RCA than using the list-of-causes (see Article V). Respectively, Cases 5-6 indicated that ARCA-tool, which uses the cause-effect diagram of the ARCA method, improves retrospectives (see Article III). However, the cases also revealed that the structured approach of RCA, including the systematic focus on target problem causes, is the most important component of the analysis. Furthermore, the results from Case 2 indicated that brainwriting combined with brainstorming is a better approach for developing action proposals than using brainstorming only.

7.2.2 Improving the ARCA method with ARCA-tool

ARCA-tool was evaluated with collocated (Case 5) and distributed (Case 6) software project retrospectives in order to study its designed support for the ARCA method (see Section 5.2.). The evaluation results regarding ARCA-tool are introduced in Section 6.2 and they contribute to the fourth research question discussed below.

RQ4: Is the developed ARCA-tool perceived as useful and easy to use in software project retrospectives applying the developed RCA method?

ARCA-tool was perceived as useful and it improves the limited ARCA method (see Section 3.5.2) in collocated and distributed software project retrospectives. The participants of Cases 5-6 perceived that ARCA-tool increases the ef-

ficiency of RCA. They also experienced that the tool is essential in distributed retrospectives. Furthermore, ARCA-tool was perceived as easy to use.

There are at least five hypotheses why ARCA-tool was perceived as useful. First, the tool enables conducting the ARCA method (see Section 5.2.), which was concluded as the key for successful software project retrospectives (see Article III). Second, the tool improves the in-depth analysis by enabling real-time visualization and simultaneous editing access to the retrospective outcome. Third, the RCA facilitator does not need to act as a scribe (see Section 4.2.2). Fourth, the tool enables conducting the ARCA method as distributed. Fifth, the tool protects the anonymity of retrospective participants.

Considering the above hypotheses, I believe that the most important success factors of the tool are that it enables conducting the ARCA method and it provides real-time visualization and simultaneous editing access to the retrospective outcome. These success factors enable conducting RCA in distributed software project retrospectives, a research problem introduced by Stålhane et al. (2003). Additionally, possible post-retrospective analyses (see articles II and IV) become easier since the retrospective findings are already electronically registered (see Section 4.2.4). Furthermore, the efficiency of analysis increases since the participants register their findings directly to the electronic cause-effect diagram (see Figure 8) instead of writing down, often illegible, Post-it notes and pasting them on a whiteboard to represent the cause-effect diagram of the target problem (Stålhane et al. 2003; Bjørnson, Wang, and Arisholm 2009).

7.3 The outcome of RCA with software project failures

A multiple case study about the ARCA method outcome was conducted in four cases of software project failures in order to evaluate whether the outcome of the ARCA method helps to express what happens, where the failures occur, and why the failures occur. The case study results contribute to the third research problem: *Does the outcome of RCA indicate how the causes of software project failures are interconnected?* Three research questions were stated for this research problem and they are answered next.

7.3.1 Frequently used process areas and cause types

The outcome of the ARCA method was analysed in Cases 1-4 in order to explain what caused software project failures at each case and where in the software development processes the causes occurred (articles II and IV). Section 6.3.1 introduces the process areas detected from the ARCA method outcome, expressing where the causes of failures occurred. Respectively, Section 6.3.2 introduces the types of causes expressing what the individual causes of failures were. Furthermore, Section 6.3.3 considers the similarities between the cases. Together these results contribute to the fifth research question discussed below.

RQ5: Which process areas and cause types were frequently used in RCA to explain software project failures?
Regarding the outcome of the ARCA method, the software project failures were commonly influenced by the problems of the management, sales & requirements, implementation, and software testing work. These process areas are similar to the ones found in software engineering process literature. Additionally, the prior studies of software project failures have emphasized these process areas (see Article IV). Furthermore, the causes of failures were commonly related to the cause types of People, Tasks, Methods, and Environment. The causes were also commonly related to the sub-types of Instructions & Experience, Values & Responsibility, Work Practices, Task Output, Task Difficulty, Existing Product, Cooperation, and Resources & Schedules. Comparison of these cause types to the results of prior studies (see Article IV) indicates that these findings are also in line with others (McLeod and MacDonell 2011).

Due to the high similarity between the case study results and prior studies, I conclude that using the ARCA method with software project failures in SME organizations helps to express where the causes of software project failures occur and what they are. In the ARCA method, the perceived causes of failures that are registered to the cause-effect diagram are expressed with rich information about their types and related process areas. Thus, analysing the outcome of the ARCA method could help to conclude what happened and where.

7.3.2 The role of bridge causes

The interconnections between the causes of software project failures were studied in order to evaluate whether the outcome of the ARCA method helps to express how the causes of software project failures affect one another. Section 6.3.4 presents the common causal relationships bridging the process areas. Furthermore, Article IV includes the in-depth analysis of the perceived causal relationships between the process areas and individual causes of software project failures at each case. Together these results contribute to the sixth research question discussed below.

RQ6: What causal relationships bridge the process areas?
The term *bridge cause* refers to a cause-effect relationship for which the process area of the effect is different than the one of its cause. Regarding the ARCA method outcome, a high number of perceived causes of software project failures in implementation, software testing, and release & deployment was bridged to the output of management work and sales & requirements. This finding consolidates the prior studies by indicating that software project failures are caused by insufficient management work and sales & requirements (see Article IV). Furthermore, the ARCA method outcome indicated that solving the problems in the management work and sales & requirements requires improvements in the implementation work and software testing too. This finding was logically compiled (see Article IV). The causal relationships between the process areas were multidirectional including three common mechanisms, bridging the process areas together. These mechanisms included *Lack of Co-*

operation, *Weak Task Backlog*, and *Lack of Software Testing Resources* (see Figure 10). Furthermore, the ARCA method outcome expressed the perceived causal relationships of individual problems local to process areas (see Article IV), and these causal relationships were also interconnected to the bridge causes. Thus, the outcome of the ARCA method helped to explain, not only the bridge causes, but the whole network of causes and effects, starting from the separated problems of software development process areas and ending with a perceived causal model of software project failures at each case.

Due to the logical relationships between the detected causes and their process areas (see Article IV), I conclude that the ARCA method helps to express how the perceived causes of software project failures are related to one another. Considering a software project failure as a problem to follow the law of causality (see Section 2.1), as indicated by Cerpa and Verner (2009), controlling the individual problems of software projects becomes important during the project. This requires knowledge about the relationships between the individual problems, i.e., the interconnections between the causes of failures. Therefore, RCA is an important part of software project retrospectives (see Figure 1). It could help to explain why the individual problems of software projects occur. Additionally, it could help to explain how these individual problems form the software project failure together.

7.3.3 Feasible targets for process improvement activities

The perceptions of practitioners and senior management on the causes for process improvement activities were studied in order to consider the importance of detecting perceived causal relationships between the causes of software project failures in software project retrospectives. The prior studies on software project failures have claimed to be important to analyse how the causes of failures are related, however, these claims are not evaluated in practice (see Article IV). Section 6.3.3 presents results on the perceived feasibility of the bridge causes for process improvement activities, and Article IV extends the results to cover an analysis of the related process areas and cause types. These results contribute to the seventh research question discussed below.

RQ7: Do the causes perceived as feasible targets for process improvement differ from the other detected causes, and if so, how?

The case study results indicate that the causes of software project failures, perceived as feasible targets for process improvement activities, are often related to the perceived causal relationships interconnecting the process areas. This means that in software project retrospectives, revealing the interconnections between the individual problems of software projects is not only theoretically reasonable (see Section 2.1), but also practically important. It leads to an understanding about the problems between software development process areas, which are important to consider in the process improvement activities (see Article IV). These results consolidate the evaluation results on the high-perceived usefulness of the ARCA method for software project retrospectives.

7.4 Implications

This thesis introduced how to use RCA in software project retrospectives and how the participants perceived its ease of use and cost-efficiency in SME organizations. It seems that software project retrospectives should use RCA. All of the evaluation results indicate that RCA is a useful part of retrospectives. These findings consolidate the prior studies that present RCA as a part of collocated retrospectives (Stålhane et al. 2003; Bjørnson, Wang, and Arisholm 2009). Additionally, our results extend the prior studies by showing that RCA is also a good approach for distributed retrospectives following the agile methods (Schwaber and Sutherland 2011).

Furthermore, the results of this thesis indicate that the focus of process improvement effort should be in the perceived causal relationships of problems. The theory of causality (see Section 2.1) consolidates this assumption. Additionally, the claim is consolidated by Card (1998) who introduced the effect of using RCA in two software organizations, and caused a total of 50% decrease in defect rates. Card's study includes significant evidence in the effect of corrective actions developed by using RCA. Together, these studies consolidate the high applicability of RCA for software process improvement activities.

Finally, there are many software engineering problems that could be considered with RCA, but they are not reported, e.g. requirement faults (Gursimran and Jeffrey 2009). Therefore, alternative work practices for the step of target problem detection should be considered. Most of the prior methods have used problem sampling, which is infeasible for unreported problems. Our results indicate that in SME organizations, problem sampling could be substituted with a focus group meeting, which makes the RCA method as lightweight and adaptable for different target problems. Such an approach could be feasible in large organizations too. However, future work is needed to consolidate this assumption.

7.5 Evaluation of the research

This section discusses the main threats to the study results. The discussion is divided into four perspectives of validity (Runeson and Höst 2008) including the construct validity, the internal validity, the external validity, and the reliability. Detailed discussion about the threats to validity can be found in the publications.

7.5.1 Construct validity

Construct validity reflects the validity of research methods used to collect the research data and draw out the conclusions regarding the research questions (Runeson and Höst 2008). The research methods used in this thesis follow the methods recommended in the framework of design science (Hevner et al. 2004) including literature reviews, field studies, multiple case studies, and controlled experiments.

The literature review about the prior RCA methods (RQ1) was structural; however, it was conducted semi-systematically (see Article I). We used predefined search words and two alternative search engines (Scopus and Google). Unfortunately, we did not keep an accurate record on the literature that we excluded. Therefore, evaluating the coverage of the review is difficult. Additionally, the list of search words was created based on our initial understanding about the relevant key words of RCA. These included "RCA", "root cause analysis", "DCA", "defect causal analysis", "defect analysis", "defect prevention", and "problem prevention". This list could have been extended with search terms including "retrospective", "postmortem", and "post-project review". These search terms were used during the latter parts of this research work to search for additional background literature (articles III and V). The found papers did not extend the set of prior RCA methods any further, which indicates that the coverage of the literature review was sufficient to make the synthesis of prior RCA methods and to develop the ARCA method.

The literature review about the prior RCA software tools (RQ2) was structural and systematic. However, the review was limited to the extensive number of hits. Respectively, the review was limited with the available information. Furthermore, the review was limited with the search term: "root cause analysis software".

The field study methods used to evaluate the ARCA method (RQ3) and ARCA-tool (RQ4) in the industrial cases and controlled student experiment creates a threat for construct validity regarding the reliability of human input. The weakness is that the evaluation results are mostly dependent on the perceptions of participants. Instead, the strength of the study is that the ARCA method and ARCA-tool were evaluated from various perspectives including the individual work practices and the retrospective outcome. Additionally, the evaluation was replicated in many different retrospective contexts and in different companies. Furthermore, the evaluation was replicated in a student experiment.

The threats to the construct validity regarding the multiple case study (RQ5-RQ7) are related to the ARCA method outcome. The outcome of RCA has been questioned, because of its high dependency on human factors (Ayad 2010). Regarding the case evaluations including interviews (see articles I and III), questionnaires (see articles I and III), and the method outcome (see Article IV), this risk is not highly significant. Furthermore, the case study results are also affected by the risks of using the grounded theory approach for analysing the ARCA method outcome regarding the process areas, cause types, interconnectedness, and feasibility for process improvement. It is possible that the interpretations about the case domains do not reflect the reality. Thus, it is possible that the classification system does not reflect the reality either. This risk decreases by the fact that we had cooperated months with the case companies. Additionally, we conducted interviews about the case domains before using the ARCA method (see articles I and IV). Therefore, our knowledge about the case domains was likely sufficient for using the grounded theory approach.

7.5.2 Internal validity

Internal validity considers the validity of the causal relationships between the studied factors and their measured effects (Runeson and Höst 2008). The study factors of this thesis include the ARCA method and ARCA-tool. The measured effects include the improvement over the existing practices in terms of the perceived efficiency and ease of use.

There is a threat to internal validity regarding the improvement of the ARCA method and ARCA-tool over the existing practices. It is possible that the researcher involvement and varying social context biased the evaluation results. It is also possible that the target problems caused bias in the study results. Section 7.5.4 considers the potential bias caused by the researcher involvement and thus it is not discussed any further in this section.

It is possible that the evaluation results regarding the existing practices were slightly biased by internal variations in the social contexts. RCA has been characterized as a "witch-hunting tool" (Latino and Latino 2006). Thus, the social context might affect the detected problems and thus decrease, or increase, the coverage of the retrospectives.

Furthermore, there is a threat to internal validity regarding the potential differences in the target problems analysed with the ARCA method and existing practices. It is possible that the target problems analysed with the ARCA method were perceived as more important than the target problems analysed with the existing practices. Regarding this risk, Cases 1-4 had tried to solve their target problems previously (Article I). Thus, they were able to provide methodological comparison without significant biases by different target problems. Respectively, the participants of Cases 5-6 (Article III) were experienced on conducting retrospectives continuously and with different types of target problems. Thus, they were able to provide methodological comparison without significant biases by monotonic target problems.

7.5.3 External validity

External validity is concerned about the generalizability of the results (Runeson and Höst 2008). The results regarding RCA and the RCA software tools are limited to the work practices of the ARCA method. Additionally, the total number of cases was only six. Furthermore, only two cases evaluated the use of ARCA-tool.

The ARCA method is based on prior RCA methods, which increases the external validity. Instead, many different work practices of the prior methods were not included in the evaluation. Respectively, we did not compare the ARCA method with the prior RCA methods. Instead, the ARCA method was compared with the existing practices of the case companies. Therefore, the external validity regarding the prior RCA methods remains somewhat low. On the other hand, the external validity regarding the existing practices is high. The evaluation covered different case contexts. Respectively, the ARCA method was evaluated with target problems at different company levels. The evaluation also covered both collocated and distributed retrospective contexts. These varia-

tions over the evaluation domains increased the external validity of the conclusions about the ARCA method. The results over the cases were remarkably similar.

Regarding the conclusions about the ARCA method outcome with software project failures including the general cause types, process areas, and the feasibility of the bridge causes for process improvement activities, the external validity is high. Cases 1-4 were used to analyse the outcome of the ARCA method in the cases of software project failures. The cases varied in terms of software project failures, case participants, and case companies. Instead, the complexity of software project failures, the use of the ARCA method, and the roles of case participants remained similar. Thus, the cases considered the causes of software project failures and their relationships from various perspectives, which collectively increase the external validity. Respectively, the similarities between the cases made the cases more comparable.

The future works should include studies with varying case contexts including projects with different size, cultural context, geographical distribution, software development methods, and software project failures. Currently our results are generalizable to SME organizations of software product companies operating in western cultures.

7.5.4 Reliability

The threats to reliability are related to potential researcher bias in the study results (Runeson and Höst 2008). There was a social tie between the researchers and the evaluation contexts. Additionally, the analyses regarding the interviews and the qualitative analysis of bridge causes are researcher dependent.

The researchers steered the use of the ARCA method in Cases 1-4 and the student experiment. This creates a threat for the reliability. It is possible that the researchers and subjects influenced one another (Sandelowski 1986). Thus, the contribution of the researchers could bias the evaluation results. This threat was controlled in Cases 5-6 (Article III). The limited ARCA method was steered solely by the company personnel. The evaluation results from Cases 1-4 are very similar to those in Cases 5-6. Thus, the potential risk of researcher bias is most likely related to the work practices of preliminary cause collection only (see Section 4.2.2), a work practice which was not included in the limited ARCA method.

Furthermore, the qualitative analyses of the interviews (articles I, III, and V) and bridge causes (Article IV) create threats for the reliability. It has been claimed to be difficult to replicate qualitative data analyses (Mays and Pope 1995). Replicating the qualitative analyses of this thesis might not be difficult, because the methods used in the data collection and analyses are clearly reported in the related articles. Instead, the potential researcher bias is related to the interpretations about the qualitative data. Regarding the interview results, triangulation (Jick 1979) of the data sources, data collection methods, and data analysis methods increase the reliability of the results. Our conclusions were based on the analyses of individual parts of research data and the analysis of all research data combined together. Such an approach has been called “her-

meneutic circle" (Klein and Myers 1999), the key principle of interpretive field study research.

Finally, regarding the qualitative analyses of the bridge causes, the classification system, including the analysis of inter-rater agreement, increases the reliability of the study results. The kappa value 0.65 indicated a good agreement between the researchers over the process area dimensions (Article IV). Thus, it is likely that the perceived causes of software project failures selected for qualitative analysis covered the bridge causes at each case. Furthermore, the bridge causes interconnecting two process areas included a relatively low number of causes. Therefore, summarizing how the process areas were interconnected was not a difficult task.

8. Conclusions and future work

This thesis made four contributions. First, a lightweight RCA method and RCA software tool was developed. Additionally, a high number of RCA methods and their work practices were introduced and discussed. Second, the use of RCA as a part of software project retrospectives was evaluated thoroughly in six industrial cases and one student experiment. Third, the use of computer facilitation during RCA was evaluated. Fourth, the outcome of RCA in the cases of software project failures was analysed.

The evaluations of RCA are limited to the work practices of the developed ARCA method, a synthesis of prior RCA methods (Article I). The use of computer facilitation is limited to the developed ARCA-tool (Article III). Furthermore, the analysis of the outcome of RCA is limited to the outcome of the ARCA method in SME organizations trying to explain why software projects have failed.

8.1 Conclusions

This is one of the first studies in the software engineering context that has systematically evaluated the perceptions of subject matter experts using RCA in their software project retrospectives. Such a systematic evaluation has not been reported before. The results indicate that RCA is an important part of software project retrospectives. It increases the efficiency of retrospectives. It is also somewhat easy to use. Additionally, it reveals feasible targets for process improvement. The evaluation covers the use of RCA in collocated and distributed software project retrospectives. Additionally, it covers the use of RCA with different target problems and various levels of retrospectives including the levels of company, organization, and team.

This is also the first study in the software engineering context that has evaluated the use of RCA software tools in distributed retrospectives. The results indicate that computer facilitation is essential for the RCA of distributed retrospectives. Respectively, RCA software increases the efficiency of collocated retrospectives. The main features of the software tool for RCA include collaborative cause-effect diagramming, corrective action development, and voting of the most important RCA outcome.

Furthermore, detailed knowledge about the actual outcome of RCA was created in this study. In the case of software project failures, the outcome of RCA helps to express hypotheses on what happened, where it happened, and why it happened. This methodological capability increases the feasibility of using

RCA as a data collection method in software project retrospectives and in the studies of software project failures.

8.2 Future work

In the future, comparative studies over the existing RCA methods and software tools should be conducted, e.g. Bjørnson et al. (2009). Conducting an in-depth analysis of software engineering problems during retrospectives is important, but also a challenging task. Retrospectives should be lightweight or they are not used (Glass 2002). Therefore, simplifying the visualization of the underlying target problem causes should be a part of future works. The high complexity and cross-functionality of software engineering problems makes it difficult to detect and analyse their causes.

It should also be studied how to simplify the causal analysis without losing the important knowledge about the solution space of target problems. A software tool could improve the causal analysis. However, the current tools require improvements.

Finally, replicative studies on the use of RCA with software project failures should be conducted in the future. Questionnaires and interviews about the causes of failures are an important part of the future studies. However, the central role of the bridge causes should be taken in account better. Bridge causes could be detected with RCA, but that requires further validation with different target problems and cases. The software engineering research has not yet filled this gap.

References

Al-Mamory, Safaa O., and Hongli Zhang. 2009. Intrusion detection alarms reduction using root cause analysis and clustering. *Computer Communications* 32 (2) (February): 419 - 430.

Álvarez, M. P. 2009. The four causes of behavior: Aristotle and skinner. *International Journal of Psychology and Psychological Therapy* 9 (1): 45-57.

Ammerman, Max. 1998. *The root cause analysis handbook: A simplified approach to identifying, correcting, and reporting workplace errors*. First Edition ed. 444 Park Avenue South, Suite 604, New York, NY 1016, USA: Productivity Press.

Andersen, Björn, and Tom Fagerhaug, eds. 2006. *Root cause analysis: Simplified tools and techniques*. Second Edition ed. United States, Milwaukee 53203: Tony A. William American Society for Quality, Quality Press.

Ayad, Amine. 2010. Critical thinking and business process improvement. *Journal of Management Development* 29 (6): 556-564.

Berander, Patrik. 2004. Using students as subjects in requirements prioritization. Paper presented at Empirical Software Engineering, 2004. ISESE'04.

Bhandari, Inderpal, Michael Halliday, Eric Tarver, David Brown, Jarir Chaar, and Ram Chillarege. 1993. A case study of software process improvement during development. *IEEE Transactions on Software Engineering* 19 (12) (December): 1157-1170.

Birk, Andreas, Torgeir Dingsøyr, and Tor Stålhane. 2002. Postmortem: Never leave a project without it. *IEEE Software* 19 (3): 43-5.

Bjarnason, Elizabeth, and Björn Regnell. 2012. Evidence-based timelines for agile project Retrospectives–A method proposal. *Agile processes in software engineering and extreme programming*: 177-184, Springer.

Bjørnson, Finn O., Alf I. Wang, and Erik Arisholm. 2009. Improving the effectiveness of root cause analysis in post mortem analysis: A controlled experiment. *Information and Software Technology* 51 (1) (January): 150 - 161.

Boh, Wai F., Sandra A. Slaughter, and Alberto J. Espinosa. 2007. Learning from experience in software development: A multilevel analysis. *Management Science* 53 (8): 1315-1331.

Burnstein, Ilene. 2003. *Practical software testing*. New York: Springer Science+Business Media.

Burr, Adrian, and Mal Owen, eds. 1996. *Statistical methods for software quality: Using metrics for process improvement*. First Edition ed. ITP A division of International Thomson Publishing Inc.

Card, David N. 1998. Learning from our mistakes with defect causal analysis. *IEEE Software* 15 (1): 56-63.

———. 1993. Defect-causal analysis drives down error rates. *Quality Time* 10 (4) (July): 98 - 99.

Carver, Jeffrey, Letizia Jaccheri, Sandro Morasca, and Forrest Shull. 2003. Issues in using students in empirical studies in software engineering education. Paper presented at Ninth International Software Metrics Symposium, 2003.

Cerpa, Narciso, and June M. Verner. 2009. Why did your project fail? *Communications of the ACM* 52 (12): 130-134.

Chillarege, Ram, Inderpal S. Bhandari, Jarir K. Chaar, Michael J. Halliday, Diane S. Moebus, Bonnie K. Ray, and Man-Yuen Wong. 1992. Orthogonal defect classification - A concept for in-process measurements. *IEEE Transactions on Software Engineering* 18 (11) (November): 943 - 956.

Collier, Bonnie, Tom DeMarco, and Peter Fearey. 1996. A defined process for project post mortem review. *IEEE Software* 13 (4): 65-72.

Cooke, David L. 2003. Learning from incidents. Paper presented at Proceedings of the 21st International Conference of the System Dynamics Society, New York, NY, USA.

Dennis, Alan R., Craig K. Tyran, Douglas R. Vogel, and Jay F. Nunamaker Jr. 1997. Group support systems for strategic planning. *Journal of Management Information Systems* 14 (1): 155-84.

Dingsøyr, Torgeir. 2005. Postmortem reviews: Purpose and approaches in software engineering. *Information and Software Technology* 47 (5): 293-303.

Dingsøyr, Torgeir, Nils B. Moe, and Øystein Nytrø. 2001. Augmenting experience reports with lightweight postmortem reviews. Paper presented at PROFES '01 Proceedings of the Third International Conference on Product Focused Software Process Improvement.

Dye, J., and T. van der Schaaf. 2002. PRISMA as a quality tool for promoting customer satisfaction in the telecommunications industry. *Reliability Engineering & System Safety* 75 (3): 303-311.

Eden, Colin. 2004. Analyzing cognitive maps to help structure issues or problems. *European Journal of Operational Research* (3): 673-686.

Edmondson, Amy C. 1996. Learning from mistakes is easier said than done: Group and organizational influences on the detection and correction of human error. *The Journal of Applied Behavioral Science* 32 (1): 5-28.

El Emam, Khaled, and A. Gunes Koru. 2008. A replicated survey of IT software project failures. *IEEE Software* 25 (5): 84-90.

Foddy, William, ed. 1994. *Constructing questions for interviews and questionnaires*. Hong Kong by Colorcraft: Cambridge University Press.

Galles, David, and Judea Pearl. 1997. Axioms of causal relevance. *Artificial Intelligence* 97 (1-2): 9-43.

Glass, R. L. 2002. Project retrospectives, and why they never happen. *IEEE Software* 19 (5) (October): 111-112.

Grady, Robert B. 1996. Software failure analysis for high-return process improvement decisions. *Hewlett-Packard Journal* 47 (4) (August): 15 - 25.

Granger, Clive WJ. 1988. Some recent development in a concept of causality. *Journal of Econometrics* 39 (1): 199-211.

Gupta, Anita, Jingyue Li, Reidar Conradi, Harald Rönneberg, and Einar Landre. 2008. A case study comparing defect profiles of a reused framework and of applications reusing it. *Empirical Software Engineering* 14 (2) (20 August): 227 - 255.

Gursimran, S. W., and C. C. Jeffrey. 2009. A systematic literature review to identify and classify software requirement errors. *Information and Software Technology* 51 (7) (July): 1087-1109.

Herbsleb, James D., and Deependra Moitra. 2001. Global software development. *IEEE Software* 18 (2): 16-20.

Hevner, Alan R., Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design science in information systems research. *MIS Quarterly* 28 (1): 75-105.

Höst, Martin, Björn Regnell, and Claes Wohlin. 2000. Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering* 5 (3): 201-14.

Hume, David. 1896. *A treatise of human nature [1739]*. reprinted from the Original Edition in three volumes and edited, with an analytical index, by L.A. Selby-Bigge ed. Oxford: Clarendon Press.

Jacobs, Jef, Jan Van Moll, Paul Krause, Rob Kusters, Jos Trienekens, and Aarnout Brombacher. 2005. Exploring defect causes in products developed by virtual teams. *Information and Software Technology* (47): 399-410.

Jacobson, I., G. Booch, and J. Rumbaugh. 1998. *The unified software development process.* Addison-Wesley.

Jalote, Pankaj, and Naresh Agrawal. 2005. Using defect analysis feedback for improving quality and productivity in iterative software development. Paper presented at Proceedings of the Information Science and Communications Technology (ICICT 2005).

Jick, Todd D. 1979. Mixing qualitative and quantitative methods: Triangulation in action. *Administrative Science Quarterly* 24 (4): 602-11.

Jin, Zhao X., John Hajdukiewicz, Geoffrey Ho, Donny Chan, and Yong-Ming Kow. 2007. Using root cause data analysis for requirements and knowledge elicitation. Paper presented at International Conference on Engineering Psychology and Cognitive Ergonomics (HCII 2007), Berlin, Germany.

Juristo, Natalia, and Ana M. Moreno. 2003. *Basics of software engineering experimentation*. London: IBT Global.

Kalinowski, Marcos, Guilherme H. Travassos, and David N. Card. 2008. Towards a defect prevention based process improvement approach. Paper presented at Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications, Parma, Italy.

Kavadias, Stylianos, and Svenja C. Sommer. 2009. The effects of problem structure and team diversity on brainstorming effectiveness. *Management Science* 55 (12) (December): 1899-1913.

Klein, Heinz K., and Michael D. Myers. 1999. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*: 67-93.

Latino, Robert J., and Kenneth C. Latino, eds. 2006. *Root cause analysis: Improving performance for bottom-line results*. Third Edition ed. 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487-2742: CRC Press.

Lee, S., J. F. Courtney, and R. M. O'Keefe. 1992. A system for organizational learning using cognitive maps. *Omega, the International Journal of Management Science* 20 (1): 23-36.

Leszak, Marek, Dewayne E. Perry, and Dieter Stoll. 2000. A case study in root cause defect analysis. Paper presented at Proceedings of the 2000 International Conference on Software Engineering.

Lethbridge, Timothy C., Susan Elliott Sim, and Janice Singer. 2005. Studying software engineers: Data collection techniques for software field studies. *Empirical Software Engineering* 10 (3): 311-341.

Livingstone, A. D., G. Jackson, and K. Priestley. 2001. Root causes analysis: Literature review. *Health & Safety Executive, Contract Research Report* 325: 1-53.

March, Salvatore T., and Gerald F. Smith. 1995. Design and natural science research on information technology. *Decision Support Systems* (15): 251-266.

Mays, Nicholas, and Catherine Pope. 1995. Rigour and qualitative research. *BMJ* 311 (8): 109-112.

Mays, Robert G. 1990. Applications of defect prevention in software development. *IEEE Journal on Selected Areas in Communications* 8 (2) (February): 164-168.

McLeod, Laurie, and Stephen G. MacDonell. 2011. Factors that affect software systems development project outcomes: A survey of research. *ACM Computing Surveys* 43 (24): 24-55.

Monteiro, Paula, Ricardo J. Machado, Rick Kazman, and Cristina Henriques. 2010. Dependency analysis between CMMI process areas. Paper presented at PROFES, LNCS 6156.

Nakashima, T., M. Oyama, H. Hisada, and N. Ishii. 1999. Analysis of software bug causes and its prevention. *Information and Software Technology* (41): 1059-1068.

Naur, P., and B. Randel. 1969. Software engineering: A report on a conference sponsored by the NATO science committee. Nato.

Pearl, Judea, ed. 2000. *Causality: Models reasoning, and inference*. United States of America: Cambridge University Press.

Rooney, James J., and Lee N. Vanden Hauvel. 2003. Collecting data for root cause analysis. *Quality Progress* 36 (11) (November): 104.

Rooney, James J., and Lee N. Vanden Heuvel. 2004. Root cause analysis for beginners. *Quality Progress* 37 (7) (August): 45 - 53.

Royce, Winston. 1970. Managing the development of large software systems. Paper presented at *Proceedings of IEEE WESCON* 26 (August).

Runeson, Per. 2003. Using students as experiment subjects–an analysis on graduate and freshmen student data. Paper presented at Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering.–Keele University, UK.

Runeson, Per, and Martin Höst. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* (14) (19 december): 131-164.

Salinger, Stephan, Laura Plonka, and Lutz Prechelt. 2007. A coding scheme development methodology using grounded theory for qualitative analysis of pair programming. Paper presented at 19th Annual Psychology of Programming Workshop, Joensuu.

Sandelowski, M. 1986. The problem of rigor in qualitative research. *ANS* 8 (3): 27-37.

Schwaber, Ken, and Jeff Sutherland. 2011. Scrum guide. *Scrum Alliance*.

Shull, Forrest, Dag I. K. Sjøberg, and Janice Singer. 2008. *Guide to advanced empirical software engineering.* Springer-Verlag London Limited.

Siekkinen, Matti, Guillaume Urvoy-Keller, Ernst W. Biersack, and Denis Collange. 2008. A root cause analysis toolkit for TCP. *Computer Networks* (52): 1846-1858.

Stålhane, Tor. 2004. Root cause analysis and gap analysis - A tale of two methods. Paper presented at EuroSPI 2004, Trondheim, Norway.

Stålhane, Tor, Torgeir Dingsøyr, Geir Hanssen, and Nils Moe. 2003. Post mortem–an assessment of two approaches. *Empirical Methods and Studies in Software Engineering*: 129-41.

Stevenson, William J., ed. 2005. *Operations management*. 8th ed. New York: McGraw-Hill/Irwin.

Svahnberg, Mikael, Aybüke Aurum, and Claes Wohlin. 2008. Using students as subjects-an empirical evaluation. Paper presented at Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement.

Terzakis, John. 2011. Virtual retrospectives for geographically dispersed software teams. *IEEE Software* 28 (3): 12-15.

Thiele, T. N. 1931. The law of causality. *The Annals of Mathematical Statistics* 2 (2): 165-169.

Traeger, Avishay, Ivan Deras, and Erez Zadok. 2008. DARC: Dynamic analysis of root causes of latency distributions. Paper presented at SIGMETRICS '08, Annapolis, Maryland, USA.

Vanhanen, Jari, Timo O. A. Lehtinen, and Casper Lassenius. 2012. Teaching real-world software engineering through a capstone project course with industrial customers. Paper presented at 1st International Workshop on Software Engineering Education Based on Real-World Experiences, EduRex 2012, Zurich.

Verner, June, Jennifer Sampson, and Narciso Cerpa. 2008. What factors lead to software project failure. Paper presented at Proceedings of Research Challenges in Information Science (RCIS 2008).

Von Zedtwitz, Maximilian. 2002. Organizational learning through post–project reviews in R&D. *R&D Management* 32 (3): 255-68.

Wang, Yingxu, and Graham King. 2000. *Software engineering processes: Principles and applications*. CRC Press LLC.

Xiangnan, L., L. Hong, and Y. Weijie. 2010. Analysis failure factors for small & medium software projects based on PLS method. Paper presented at The 2nd IEEE International Conference on Information Management and Engineering (ICIME 2010).

Yin, Robert K., ed. 1994. *Case study research: Design and methods*. 2nd Edition ed. United States of America: SAGE Publications.

Root cause analysis is a systematic process for detecting a target problem and recognizing its root causes. This dissertation includes the development and evaluation of a lightweight root cause analysis method (ARCA) and software tool (ARCA-tool) in the retrospectives of small- and medium-sized global software organizations. The evaluation results from a total of six industrial cases and one controlled experiment are all positive and they indicate that the ARCA method is cost-efficient and easy-to-use for detecting the causes of software project failures. This finding makes an important contribution to the methodologies of software project retrospectives. When considering the causes of failures, we should analyze what happened and where it happened, but also why it happened. Root cause analysis is a lightweight retrospective approach to such data collection and analysis.

Aalto-DD 159/2014

ISBN 978-952-60-5907-5 (printed)
ISBN 978-952-60-5908-2 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science and Engineering
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS