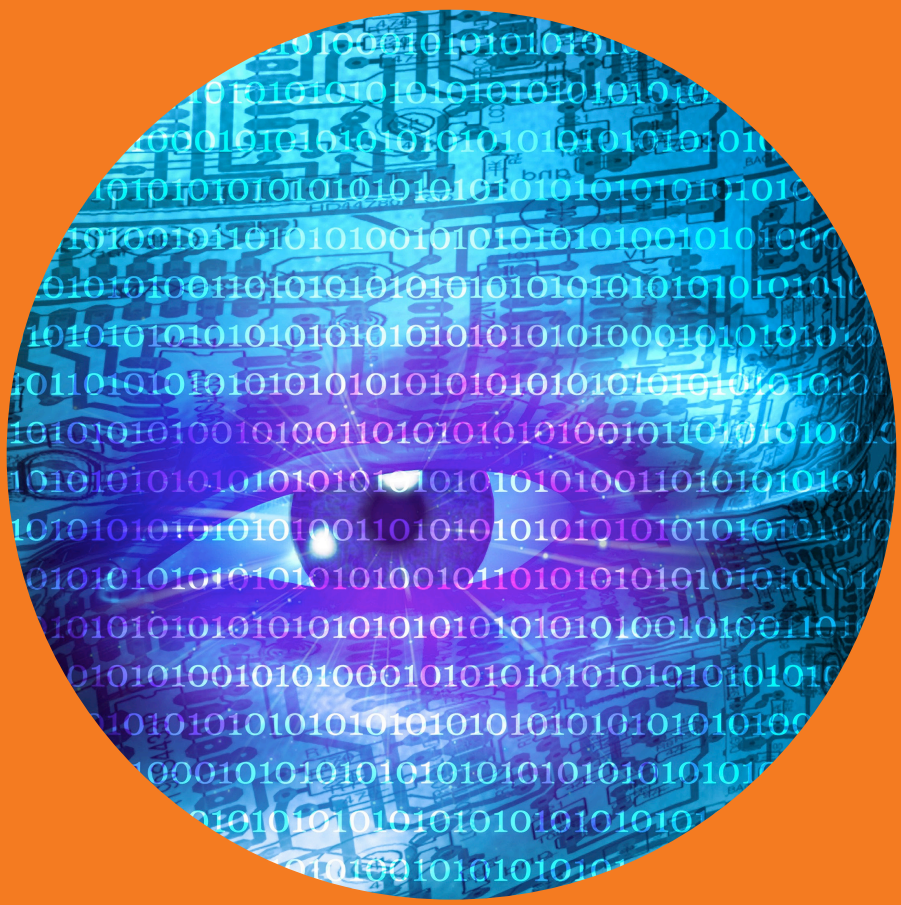
Department of Computer Science and Engineering

Empirical studies on exploratory software testing

Juha Itkonen



Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 107/2011

Empirical studies on exploratory software testing

Juha Itkonen

Doctoral dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the School of Science for public examination and debate in Auditorium T1 at the Aalto University School of Science (Espoo, Finland) on the 18th of November 2011 at 12 noon.

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

Supervisor
Professor Casper Lassenius

Preliminary examiners
Professor Natalia Juristo, Universidad Politécnica de Madrid, Spain
Professor Markku Oivo, University of Oulu, Finland

Opponent
Professor Magne Jørgensen, Simula Research Laboratory, Norway

Aalto University publication series
DOCTORAL DISSERTATIONS 107/2011

© Juha Itkonen

ISBN 978-952-60-4339-5 (pdf)
ISBN 978-952-60-4338-8 (printed)
ISSN-L 1799-4934
ISSN 1799-4942 (pdf)
ISSN 1799-4934 (printed)

Unigrafia Oy
Helsinki 2011

Finland

The dissertation can be read at http://lib.tkk.fi/Diss/

Author
Juha Itkonen

Name of the doctoral dissertation
Empirical studies on exploratory software testing

Publisher School of Science

Unit Department of Computer Science and Engineering

Series Aalto University publication series DOCTORAL DISSERTATIONS 107/2011

Field of research Software engineering

Manuscript submitted 14 June 2011 **Manuscript revised** 26 September 2011

Date of the defence 18 November 2011 **Language** English

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

Abstract

Exploratory software testing (ET) is a practically relevant approach to software testing that lacks scientific knowledge. In ET, the tester's work is not based on predesigned and documented test cases. Instead, testing is guided by a higher-level plan or mission, and the testing work involves parallel test design, test execution, and learning. One of the distinct characteristics of ET is that the tester designs the tests during ET and uses information gained to design new and better tests continuously. The ET approach relies on testers' skills and experience. The main claimed benefits of ET are the tester's ability to apply personal knowledge and creativity during testing as well as effectiveness, efficiency, and agility in terms of adapting to changes and working with imperfect documentation.

In this thesis, the ET approach has been studied using empirical research methods. Two case studies, one controlled experiment, and two field studies were performed to address three research goals: defining ET and understanding its applicability based on the literature; empirically investigating the benefits and shortcomings of ET; and providing empirically based results on how the ET approach is applied in practice.

This research identifies different approaches to ET in industry and describes concrete testing practices. The role of the tester's personal knowledge is identified in the literature, and this research provides a detailed analysis of the application of personal knowledge in failure detection using ET.

The main conclusions of this work are that ET can be as effective as test case-based approaches and even more efficient in certain contexts. The testers are capable of utilizing their personal knowledge in failure detection, and the role of personal knowledge is important in the ET approach. In addition, software testing in product organizations seems to involve multiple diverse organizational groups, and ET was found to be an applicable approach to engage domain experts in testing.

The main implications of this thesis are introducing the exploratory testing approach to the research community and motivating its relevance by providing empirical studies in industry. In addition, the results of the effectiveness and efficiency of ET as well as the qualitative data on exploratory testing practices and the detailed analysis of knowledge in exploratory testing work are valuable for the research community. The main practical implications include presenting the benefits and applicability of the ET approach along with the potential shortcomings and providing empirical evidence regarding the benefits of ET.

Keywords software testing, exploratory testing, defect detection, effectiveness, experience, domain knowledge, case study, controlled experiment, field observation

ISBN (printed) 978-952-60-4338-8 **ISBN (pdf)** 978-952-60-4339-5

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

Location of publisher Espoo **Location of printing** Helsinki **Year** 2011

Pages 178 **The dissertation can be read at** http://lib.tkk.fi/Diss/

Aalto-yliopisto

Tiivistelmä

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

Tekijä
Juha Itkonen

Väitöskirjan nimi
Empiirisiä tutkimuksia tutkivasta ohjelmistotestauksesta

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietotekniikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 107/2011

Tutkimusala Ohjelmistotuotanto

Käsikirjoituksen pvm 14.06.2011 **Korjatun käsikirjoituksen pvm** 26.09.2011

Väitöspäivä 18.11.2011 **Kieli** Englanti

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

Tiivistelmä

Tutkiva testaus on merkittävä manuaalisen ohjelmistotestauksen lähestymistapa, jota ei ole aiemmin tutkittu tieteellisesti. Se ei perustu etukäteen dokumentoituihin testitapauksiin vaan rinnakkaiseen testien suunnitteluun, suorittamiseen ja oppimiseen. Testaaja suunnittelee testejä testauksen aikana ja hyödyntää saamaansa informaatiota uusien, parempien testien kehittämisessä. Tutkiva testaus perustuu testaajan taitoihin ja kokemukseen. Sen väitettyjä hyötyjä ovat kustannustehokkuus sekä mahdollisuus hyödyntää testaajan tietämystä ja luovuutta. Tutkivaa testausta voidaan pitää myös ketteränä lähestymistapana, sillä se sietää muuttuvaa ja epätäydellistä dokumentaatiota.

Tässä väitöskirjassa tutkivan ohjelmistotestauksen lähestymistapaa on tutkittu empiirisillä tutkimusmenetelmillä. Työssä on suoritettu kaksi tapaustutkimusta, yksi kontrolloitu koe ja kaksi havainnointitutkimusta teollisuudessa. Tutkimuksella on ollut kolme tavoitetta: tutkivan ohjelmistotestauksen määrittely ja sen soveltuvuuden arviointi kirjallisuuden perusteella, hyötyjen ja puutteiden empiirinen evaluointi ja empiiristen tulosten tuottaminen tutkivan ohjelmistotestauksen soveltamisesta käytännön ohjelmistokehitystyössä.

Tässä työssä on tunnistettu monia tapoja, joilla tutkivaa testausta sovelletaan ohjelmistoteollisuudessa, ja kuvattu myös konkreettisia testauskäytäntöjä. Testaajan tietämyksen merkitys on tunnistettu kirjallisuudessa, ja tässä työssä kuvataan yksityiskohtainen analyysi siitä, mikä on tietämyksen merkitys ohjelmistovikojen tunnistamisessa tutkivaa testausta käytettäessä.

Tämän tutkimuksen tärkeimmät havainnot ovat, että tutkiva ohjelmistotestaus voi joissain tilanteissa olla yhtä tehokas ja kustannuksiltaan edullisempi kuin testitapauksiin perustuva testaus. Testaajat pystyvät hyödyntämään tietämystään vikojen havaitsemisessa, ja tietämyksen vaikutus tutkivassa testauksessa on merkittävä. Lisäksi ohjelmistotestaukseen osallistuu tuoteorganisaatioissa ihmisiä monista erilaisista ryhmistä, ja tutkiva testaus on toimiva tapa hyödyntää näitä eri sovellusalueen osaajia testauksessa.

Tutkimuksen tärkeimmät vaikutukset ovat tutkivan ohjelmistotestauksen lähestymistavan esille nostaminen tutkimusyhteisössä ja sen motivointi empiirisen tutkimuksen avulla. Lisäksi tutkimusyhteisölle merkittäviä ovat työn tulokset kustannustehokkuudesta, kvalitatiiviset tulokset käytännöistä sekä huolellinen analyysi tietämyksen merkityksestä. Käytännöllisesti arvokkaita ovat ymmärrys tutkivan ohjelmistotestauksen soveltuvuudesta ja hyödyistä suhteessa puutteisiin sekä empiiriset tulokset sen kustannustehokkuudesta.

Avainsanat ohjelmistotestaus, tutkiva testaus, vikojen havaitseminen, tehokkuus, kokemus, sovellusalueen tuntemus, tapaustutkimus, kontrolloitu koe, havainnointitutkimus

ISBN (painettu) 978-952-60-4338-8 **ISBN (pdf)** 978-952-60-4339-5

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

Julkaisupaikka Espoo **Painopaikka** Helsinki **Vuosi** 2011

Sivumäärä 178 **Luettavissa verkossa osoitteessa** http://lib.tkk.fi/Diss/

Acknowledgements

I want to thank many people who have contributed to the research and made this thesis possible. First, I would like to thank my supervisor, Casper Lassenius, for his continuous support and guidance in my research work. He has always supported and encouraged me to pursue my work on this rather uncommon research topic. I am grateful for his persistence, as he never lost his faith in my work during these years.

This research has been a team effort with my dear researcher colleagues at SoberIT, not a lonely journey. I want to express special thanks for their co-operation, help, and encouragement to my closest friends and colleagues in the Software Process Research Group: Mika Mäntylä, Kristian Rautiainen, Jari Vanhanen, Jarno Vähäniitty, and Timo Lehtinen.

Many people have participated in the arrangements and data collection for this research. I want to thank Mikko Rusama for his help with the experiment arrangements and the anonymous students and software development professionals who have participated in my research as subjects.

I want to thank my pre-examiners, Professor Natalia Juristo and Professor Markku Oivo, for helpful comments. In addition, I thank Professor Claes Wohlin and other SERL researchers at Blekinge Institute of Technology for the opportunity to spend three months as their guest and for their support while I was finalizing this dissertation.

This research would not have been possible without the financial support of Finnish Funding Agency for Technology (Tekes), Graduate School for Electronic Business and Software Industry (GEBSI), and Graduate School on Software Systems and Engineering (SoSE). Particularly important have been the contributions of the participating companies of the SHAPE and ESPA research projects in providing access to their development organizations and empirical data.

The most valuable support I have received came from my family. I thank my wife Kati for her love and support. I will always remain grateful to her for bearing the sole responsibility for our home and children while I was away. Finally, spending cheerful moments with our two wonderful children, Iiro and Riia, has helped me to put things in the right perspective during this work.

Juha Itkonen
Espoo, October 2011

List of publications

I **Toward an Understanding of Quality Assurance in Agile Software Development**
Juha Itkonen, Kristian Rautiainen, and Casper Lassenius
Published in *International Journal of Agile Manufacturing*, 2005, vol 8, no. 2: 39–49.

II **Exploratory Testing: A Multiple Case Study**
Juha Itkonen and Kristian Rautiainen
Published in *Proceedings of International Symposium on Empirical Software Engineering*, 2005, pp. 84–93.

III **Defect Detection Efficiency: Test Case Based vs. Exploratory Testing**
Juha Itkonen, Mika V. Mäntylä, and Casper Lassenius
Published in *Proceedings of International Symposium on Empirical Software Engineering and Measurement*, 2007, pp. 61–70.

IV **How Do Testers Do It? An Exploratory Study on Manual Testing Practices**
Juha Itkonen, Mika V. Mäntylä, and Casper Lassenius
Published in *Proceedings of International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 494–497.

V **The Role of Knowledge in Failure Detection During Exploratory Software Testing**
Juha Itkonen, Mika V. Mäntylä, and Casper Lassenius
Submitted to *IEEE Transactions on Software Engineering*, May 2011, 17 pages.

VI **Who Tested My Software? Testing as an Organizationally Cross-Cutting Activity**
Mika V. Mäntylä, Juha Itkonen, and Joonas Iivonen
Published in *Software Quality Journal*, published online 21st August 2011, 28 pages.

Author's contributions

In articles I–V, the author of this thesis was the principal author and was responsible for the research idea, research design, data collection, and writing the articles. The author has performed all the data analyses and written, or co-written in the case of article II, all sections of the articles. In article II, writing the article and the final data analysis cycle were collaborative activities together with the second author. In article VI, the author of the thesis contributed to the original ideas and research design and helped in data col-

lection. The author's biggest contribution in article VI was in the writing and analysis phase, where he improved the related work, improved and extended the analyses, and restructured and improved the results and discussion for the article's final form.

Table of Contents

Part I: Summary

1. Introduction

"One important outcome of a test process is a better, smarter tester."
—Lesson 46 in Kaner et al. (2002)

Software testing is commonly seen as a mechanical, dull, and repetitive activity that involves substantial human labor and should be automated as much as possible. This paradigm of software testing is strongly visible in the research literature, but practitioners, testing professionals and consultants, often see testing in a remarkably different light. The above quote from Kaner et al. (2002) powerfully illustrates this approach to software testing. Software testing practitioners might describe manual testing as an intellectually challenging, creative, and professionally demanding task that requires a wide variety of knowledge and skills. Exploratory software testing is such a testing approach that relies on the tester's skills, knowledge, and expertise instead of detailed test documentation. In this thesis, the exploratory software testing approach is studied.

1.1 Motivation and background

Software testing is a fundamental practice needed in software engineering to ensure good enough quality of software products. Testing aims at improving software quality through revealing software faults early enough to be fixed before release to customers and users. Traditionally, in the literature and especially in research, software testing has been described through a document-driven and prescriptive test case-based testing (TCBT) paradigm. In the testing literature, the term "testing technique" is very clearly interpreted as the test case design or generation technique. Most of the research, see e.g. (Juristo et al. 2004), is focused on techniques for test case design, selection, prioritization, and optimization. The results, however, do not show much difference between the various testing techniques and, e.g., random testing (Juristo et al. 2009). The development of test automation has improved testing in many ways, and test automation has become increasingly popular because of approaches such as test-driven development

and extreme programming (Beck 1999, Janzen & Saiedian 2005). Many authors, however, have pointed out that automated testing cannot replace manual testing because most new defects are found by manual testing even when automated testing is applied. Thus, instead of viewing test automation as a replacement for manual testing, it is usually viewed as a way of removing the enactment of simple and repetitive tasks from human testers in order to free up time for more intelligent manual testing (Fewster & Graham 1999, Andersson & Runeson 2002, Berner et al. 2005). Automating software testing involves several severe practical challenges, including the costs and maintainability of the tests and the tests' ability to reveal new defects when rerun (Fewster & Graham 1999, Persson & Yilmaztürk 2004, Berner et al. 2005). Because of the benefits of manual testing and the challenges of test automation, it is unlikely that manual testers will be replaced by automated testing in the near future.

Manual testing approaches are still a highly relevant part of most software development efforts, especially in the context of interactive systems with people as users.

> "Manual testing is the best choice for finding bugs related to the underlying business logic of an application. Business logic is the code that implements user requirements; in other words, it is the code that customers buy the software for. Business logic is complex and requires a human in the loop to verify that it is correct, a task that automation is too often ill-suited to accomplish." (Whittaker 2009)

Human testers seem to have many benefits over automated approaches that make manual testing an effective approach. These characteristics of humans as testers include creativity, intelligence, the ability to efficiently recognize a variety of problems, and domain knowledge. Knowledge of the application domain, users, and how they work with the system is recognized as a significant factor in software testing both by practitioners (Engelke & Olivier 2002, Kharlamov et al. 2008) and researchers (Beer & Ramler 2008, Kettunen et al. 2010).

Interestingly, despite its high relevance to practical software development, manual testing and humans as testers have not been extensively studied in the software engineering community. Defect detection activity (i.e., a human tester's behavior and activities when recognizing software failures during test execution) is a virtually unstudied area. Since one of the most important goals of software testing is to reveal defects in the software, test execution and defect detection activity deserve more research. After all, the defects are found during the test execution.

An early motivator for the research reported in this thesis was the hype surrounding agile software development methods a few years after the millennium and the experiences of challenges in organizing testing and other quality practices in agile development contexts. At that time, the exploratory testing (ET) approach also gained more attention in the practitioner literature, and it seemed a good match with the agile values and principles.

The exploratory approach to software testing has been recognized in the literature for decades but has gained more attention in the practitioner literature since 2000. The fundamental principles of ET are, first, relying on testers' knowledge and skills by testing without using detailed and prescribed test cases and, second, parallel learning, test design, and test execution. Practitioner reports on ET have proposed that, in some situations, it could be even orders of magnitude more efficient than TCBT (Bach 2004). Other claimed benefits of ET include the ability to better utilize testers' creativity, experience, and skills; lower documentation overhead; and lower reliance on comprehensive documentation (Bach 2000, Kaner et al. 2002, Våga & Amland 2002, Lyndsay & van Eeden 2003, Bach 2004). The ET approach builds on a human tester's strengths and focuses on test execution and revealing relevant defects instead of on documentation, repeatability, or coverage.

Considering the claims and practitioner reports of ET and the identified role of experience and knowledge in defect detection and software testing, the experience-based and exploratory paradigms of software testing deserve more dedicated research efforts. In this thesis, the ET approach is studied both in realistic industrial and more controlled academic settings. The results illustrate the benefits and shortcomings of ET and deeper qualitative knowledge on how testers work, identify failures, and apply their knowledge in practice when performing exploratory software testing activities.

> "The world of exploratory manual testing is one of the most challenging and satisfying jobs in the IT industry." (Whittaker 2009)

1.2 Research goals

The goals of this research are derived from the practical notion of the gap between the theory and practical reality of software testing. Testing literature and theory are hard to relate to the pragmatic approaches to testing that development organizations apply in practice. The ET approach that can be found in the practitioner literature seems to share similar values that are applied in industrial practice. The ET approach has not attracted much interest by the software engineering research community. The high-level re-

search problem in this thesis is: *How do exploratory software testing approaches work in practice?*

This high-level problem is addressed in this thesis through three goals:

Goal 1: Define ET and understand the applicability of ET based on the literature.
Goal 2: Empirically investigate the benefits and shortcomings of ET.
Goal 3: Provide empirically based results on how the ET approach is applied in practice.

This thesis and the presentation of the results and discussion are structured around these three goals. The goals are described and more detailed research questions are presented in Section 3.1.

1.3 Structure of the thesis

This thesis consists of a summary part and the research articles. The summary part involves a brief review of relevant related work in Section 2, and the research goals, questions, and methodology are introduced in Section 3. The results of this work, structured by the research goals, are presented in Section 4. The results are discussed in Section 5, including answers to the research questions, limitations, and implications of this work to research and practice. Finally, conclusions and directions for future work are stated in the last section of the summary part. Part II includes the six original research articles.

2. Related work

In this section, I review the relevant previous research and literature that relates to this thesis work. The related work is covered in two parts. First, the literature covering exploratory software testing approach is presented, including work on ET approaches, reported practitioner experiences of ET, and scientific research on ET. Second, the existing studies on the role of experience and knowledge in software testing and defect identification are reviewed.

2.1 Exploratory software testing

In this section, the relevant literature on ET is reviewed. Scientific research on ET is still scarce and was practically non-existent prior to this thesis work. Thus, part of this literature review has to rely on practitioner reports and books. This section is divided into four subsections. First, the ET approach is introduced and described based on books and the practitioner literature. Second, practitioner reports on experiences and claimed benefits are covered, and, finally, scientific research on ET is reviewed.

2.1.1 Exploratory software testing approach

The term exploratory testing was first used in software testing books by Kaner et al. (1999). The exploratory software testing approach has been acknowledged in software testing books since the 1970s (Myers 1979), but has been referred to mostly as an ad hoc approach or error guessing without any concrete description of how to perform such testing.

In most of the sources, ET is seen as a useful and effective approach to testing but as a complementary approach to structured and systematic test case-based (TCBT) techniques. Exploratory testing can be viewed as a different way of applying testing techniques and theories. Exploratory testing can utilize both the same and different techniques as the traditional predesigned testing. It can be stated that ET is a more efficient way of applying not only similar testing theories and strategies that are used for predesigned testing but also the tester's skills, experience, and tacit knowledge.

In ET, the result of applying the testing strategies is not written down in detail beforehand, and the test design is not used as a restrictive script during the test execution.

There is not much literature on exploratory testing, but it has been well-covered in books by Kaner et al. (2002) and, more recently, by Whittaker (2009). Many other books on testing also cover ET to some extent, see, e.g., (Craig & Jaskiel 2002, Whittaker 2003, Copeland 2004, Page et al. 2008, Crispin & Gregory 2009). James Bach (2004) described it in more detailed writings, and Tinkham and Kaner (2003a) covered the need for questioning skills and the heuristic nature of ET. Jonathan Bach (2000) published the first approach to managing ET, called "session-based test management." ET was defined in SWEBOK as:

> "*Exploratory testing is defined as simultaneous learning, test design, and test execution; that is, the tests are not defined in advance in an established test plan, but are dynamically designed, executed, and modified.*" (Abran et al. 2004).

Bolton (2005) gave an explanation and examples of how to explore without specifications. Whittaker (2009) published the first dedicated ET book giving a detailed description of a tour-based exploratory testing approach. In addition, Bach (1999) described an exploratory testing procedure for testing the functionality and stability of a software application for the "Certified for Microsoft Windows Logo."

Exploratory software testing seems to align well with the values and principles of agile software development (Fowler & Highsmith 2001, Cockburn 2002). Originally, the exploratory testing approach was not recognized as part of agile methods; e.g., Crispin and House (2003) did not allow manual testing at all in their first agile testing book. Also, in the descriptions of ET, only hints of applicability in the agile context could be found, e.g., (Bach 2004). This lack of coverage of the testing practices in agile development or the applicability of the exploratory testing approach in the agile development context was the motivation to perform the literature-based analysis presented in article I of this thesis.

In the context of open-source software, the testing approach is typically informal and exploratory in nature, not controlled by strict test case documentation. Instead, testing in OSS is based on a large number of volunteer testers performing free-form testing in numerous different locations and software and hardware environments (Aberdour 2007). Successful open-source software projects are one example of the potential abilities of the exploratory testing approach, demonstrating that the prescribed test-case-

driven approach is not the only possible way of detecting important defects and achieving a high-quality software system.

2.1.2 Practitioner reports on ET

Practitioners and consultants often describe ET as the way in which experienced testers work; for example, Lyndsay and van Eeden (2003) wrote, "Session-based testing mirrors the activities of experienced testers, but is not the subject of a great many papers or books."

Practitioner reports of experiences of applying exploratory testing approaches in industry have claimed that ET is effective in detecting defects as well as cost-efficient. Bach suggested that, in some situations, ET can be much more efficient than scripted testing (Bach 2004).

The most concretely described exploratory testing approach in the practitioner literature is session-based testing (SBT) (Bach 2000, Lyndsay & van Eeden 2003, Wood & James 2003). SBT is a method for managing exploratory testing that enables planning and tracking ET without sacrificing the strengths of the exploratory approach. At the heart of SBT are strictly restricted, time-boxed testing sessions instead of test cases as the unit of testing, and the testing work is planned and controlled on the granularity of the sessions. A typical length of a testing session is from a few hours to half a day at maximum. Within the limits of a session, the tester's activities are not strictly controlled or predesigned; however, these sessions are directed and planned, which enables the management of testing efforts on a higher-than-test case level.

The first documented ET method was session-based test management (SBTM), first described in a magazine article by Bach (2000). In SBTM, as in session-based testing in general, the basic work unit of testing is a test session, not a test case. The method includes practices for planning, managing, reporting, and tracking the progress of exploratory testing as short sessions, but it does not describe the actual testing techniques or defect detection strategies to be used in the testing sessions. The basic building blocks of SBTM are a charter, time-boxed sessions, reviewable results, and a debriefing.

Another, more detailed case report of performing exploratory SBT was published on the Internet by Lyndsay and van Eeden (2003). In their approach, Lyndsay and van Eeden introduced the concept of "test points" for controlling the scope and coverage of testing. Test points describe the testing tasks that are performed during sessions. A test point can be described as a unit of work and a test session as a unit of time, which separate the

concepts of the test session and the testing task more clearly than in Bach's SBTM.

Lyndsay and van Eeden (2003) reported many results and lessons learned after introducing their SBT approach in the case organization. The most important results were the ability to measure and control the exploratory testing process and the visibility of the testing work to the test manager. The testing team also felt in control of their work. They could see the size of the work, current status, and progress. The improved visibility of the testing process improved the trust between the developers and the testing team. Finally, they reported that the tested product in this case was more stable and had fewer outstanding defects.

Wood and James (2003) reported the benefits of applying exploratory SBT in the medical software domain. They identified several problems compromising the effectiveness of testing in medical software testing, including highly compartmentalized testing, excess "housekeeping" documentation at the expense of actual testing, an emphasized focus on requirements and code coverage instead of defect discovery, and repetitive testing. As a solution to these problems, Wood and James proposed complementing the standard testing methods with exploratory SBT. They described an SBT approach for the medical device software domain and reported lessons learned from using STB. They found STB to be an effective testing approach, especially if the testers were independent of the development and V&V activities. They suggested that exploratory SBT is best applicable in areas where heavy user interaction and outcomes can be confirmed quickly.

Våga and Amland (2002) reported a case description involving the application of an exploratory testing approach in a very tightly scheduled acceptance testing project. The tested software was a proprietary Web publishing system for a large multinational IT company. The documentation of the system was scarce, and the developers still changed and fixed the system frequently, even though it was going into production. They combined exploratory and pair testing in their two-day testing project. The reported project was an extreme case where the task was to test a completed Web system in two days using inexperienced testers, mainly end users, who had only two days of training in ET prior to the testing work. They had 14 testers who, during this two-day acceptance testing, reported approximately 150 defects. It took 3 months to fix the 65 most critical of the found defects before taking the system into production with limited functionality. The test team managed to stop a system with too many defects from going into production.

2.1.3 Scientific ET research

In the scientific research literature, only a few works even related to ET have been published prior to this thesis work. There is a body of empirical research on the effectiveness of numerous test case design techniques (Juristo et al. 2004) but no studies comparing the ET and test case-based approaches. In a related study, Houdek et al. (2002) studied defect detection effectiveness in an executable specification context, comparing systematic testing and experience-based ad hoc simulation. According to their results, ad hoc simulation required less effort than the systematic techniques of inspection and testing, and there was no difference in effectiveness (Houdek et al. 2002). This gives some support to the hypothesis that freestyle ET could be an efficient approach for detecting defects. However, it is hard to generalize their findings from the executable specification context to functional software testing. Later, do Nascimento and Machado (2007) compared exploratory and model-based feature testing in the mobile phone application domain. They found ET approach to be as effective as and even more efficient than the other testing approaches. The results were, thus, similar to those previously reported by Houdek et al.

Other studied aspects of ET include the effect of the individual characteristics on ET. Tinkham and Kaner (2003b) used differences in learning styles to explain different styles of testing and, more recently, Shoaib et al. (2009) studied the effect of personality traits on exploratory testing performance and found that extroverted personality types might be more likely good at exploratory testing.

Tuomikoski and Tervonen (2009) reported good experiences in integrating exploratory testing sessions with the agile Scrum methodology. They reported good results in terms of the number of revealed defects as well as the benefits of sharing knowledge and forming a common understanding of the actual quality level through the exploratory testing sessions (Tuomikoski & Tervonen 2009).

These few studies on ET do not provide any conclusive results or enable more detailed synthesis but give some support to the hypothesis that ET could be an effective and efficient testing approach in certain contexts.

Another related research area that has emerged during recent years is empirical research on software testing in real-world contexts. Some of such studies are relevant from the ET point of view. Pichler and Ramler (2008) applied ET in testing a highly interactive GUI editor and developed software tools to support exploratory GUI testing. Researchers have also determined that the ET approach seems to match well with agile development processes; e.g., Tuomikoski and Tervonen (2009) described positive experiences in

using team exploratory testing sessions as part of the agile Scrum development process. Martin et al. (2007) gave a detailed description of a "systems integration testing" approach that is highly exploratory in their ethnography of testing at a small agile company. Kasurinen et al. (2010) observed exploratory testing as part of a more generic risk-based approach to testing.

In addition to refereed scientific forums, multiple academic theses have recently been published on the ET topic, which gives the impression that ET is gaining more attention among academics. See, e.g. (Bhatti & Ghazi 2010, Hellmann 2010, Hulkkonen 2010, Naseer & Zulfiqar 2010, Saukkoriipi 2010, Shah & Alvi 2010).

As a summary of the review of related ET work, it can be stated that exploratory testing has been promoted by practitioners and positive experiences with the applicability and effectiveness of ET have been presented in the practitioner literature. Research on ET in scientific forums is emerging. There are some results of comparing ET with other testing approaches that support the effectiveness and efficiency of the ET approach. There is a lack of studies focusing on actual exploratory testing practices and activities.

2.2 The role of experience and knowledge in software testing

In this section, the research literature on the role of personal experience and knowledge in the context of software testing and defect identification is reviewed. The section is divided into two subsections. First, research on the effect of experience and domain knowledge in software testing is reviewed. Second, the relevant literature on the oracle problem and defect identification is discussed.

2.2.1 Effect of experience and domain knowledge

In industrial practice, it is common that some part of the test case selection and design as well as evaluation of the expected outcome is left to the individual testers to do based on experience and tacit knowledge instead of rigorously documented techniques and outcomes. Studies on industrial practice report that rigorous, systematic, and thoroughly documented TCBT is not dominating testing practices in industry (Andersson & Runeson 2002, Ng et al. 2004, Runeson 2006, Engström & Runeson 2010). It seems that an experience-based approach is an important complementary part of testing even in contexts where rigorous and systematically documented testing is required. Practitioners have proposed an experience-based testing approach, e.g., in the financial (Kharlamov et al. 2008) and medical (Engelke & Olivier 2002, Wood & James 2003) domains. According to some re-

search, the lack of rigorous documentation is not always considered a problem in industry (Andersson & Runeson 2002), and there are significant challenges in keeping documentation up to date (Forward & Lethbridge 2002). Some practitioners even consider a too-rigorous approach to test documentation harmful (Dallas 2010). Although there are limited empirical studies on the industrial practice of software testing, these findings give some insight into the practical relevance of the experience-based approach to software testing.

Expertise has been studied in the design (Cross 2004), software engineering (Turley & Bieman 1995), and software design contexts (Adelson & Soloway 1985, Sonnentag 1998). Outside software engineering expertise and competence have been studied widely; e.g., Sandberg (2000) studied human competence at car engine optimizing work. However, scientific research on the role of experience and knowledge in the software testing context is still rare. Only a few studies have been published focusing specifically on the role of experience in software testing (Beer & Ramler 2008, Kettunen et al. 2010, Merkel & Kanij 2010, Poon et al. 2011).

Beer and Ramler (2008) studied the role of experience in three case studies and described the role of experience in the development of test cases, regression testing, and test automation. They found that domain knowledge, in addition to testing knowledge, is crucial in testing. They indicated that the typical knowledge development path of senior testers started with strong domain knowledge. Testing experience, was gained later through working in testing, seminars, and working with external consultants. They concluded that "test design is to a considerable extent based on experience and experience-based testing is an important supplementary approach to requirements-based testing" (Beer & Ramler 2008). In all three cases of Beer and Ramler, the test cases were designed before the actual testing, which means that their research did not give any insight into the exploratory testing approach. Kettunen et al. (2010) also reported domain knowledge to be the most emphasized area of testers' expertise in their study but emphasized also the role of technical knowledge, especially in the agile context.

Merkel and Kanij (2010) performed a survey on the effect of experience and individual differences in software testing. The survey results showed that testing practitioners considered both testing and domain experience as important factors affecting testing performance. Testing specific training or certification was not found to be as important, but the individual traits of a tester were found to be highly influential on tester performance by the respondents (Merkel & Kanij 2010).

Poon et al. (2011) investigated experimentally the differences in the types and amounts of mistakes made in test case identification between inexperienced and experienced software testers. They also studied the reduction of defect mistake rates when providing testers with an identification checklist. They found large variations among individual subjects, especially in the case of inexperienced subjects. Experienced subjects identified more test categories and made fewer mistakes, and, in particular, the number of missing categories in complex cases was considerably lower for experienced subjects. However, Poon et al. concluded that experienced testers are not necessarily better than inexperienced ones in every respect. Experienced ones made more mistakes of certain types. In addition, the contribution of experience to performance decreases when the complexity of the tested functionality increases. Using a checklist reduced the number of missing categories and all types of mistakes (Poon et al. 2011).

The existing research on the role and effects of experience in software testing raises the hypothesis that experience has an important effect on testing performance and that domain knowledge will be even more important than testing experience.

2.2.2 Oracle problem and defect identification

The "test oracle" is a concept describing a method that is used to recognize correct and incorrect test output during software testing (Howden 1978, Beizer 1990, Baresi & Young 2001, Abran et al. 2004). This defect recognition is one of the most crucial activities in testing, and the existence of a test oracle is recognized as a fundamental requirement in all kinds of testing (Howden 1978, Whittaker 2000, Baresi & Young 2001, Memon et al. 2003). The challenge of finding such a reliable oracle for testing is referred to as "the oracle problem" in the literature. The oracle problem is often investigated in the context of test automation in striving for automated test oracles (Baresi & Young 2001, Abran et al. 2004, Shahamiri et al. 2011). While the same problem is associated with all software testing, in manual testing, the TCBT paradigm aims at solving the oracle problem by predefining the expected result in detail: "In real testing *the outcome is predicted and documented before the test is run*" (Beizer 1990). This assumption of the existence and availability of a test oracle is referred to as the "oracle assumption" in the literature (Howden 1978, Weyuker 1982).

In practice, however, requirements, specifications, and test cases are seldom perfect in terms of comprehensiveness and accuracy. Weyuker (1982) described her experiences in testing what she called non-testable programs. She recognized the common challenge that a reliable oracle does not always

exist or is not practically available for testers. Even in these situations, testing is possible, and Weyuker described several strategies for testing such programs. The strategies included a partial oracle, a pseudo-oracle, and using simplified data (Weyuker 1982). In referring to a partial oracle, Weyuker (1982) meant a situation in which the "tester is able to state with assurance that a result is incorrect without actually knowing the correct answer." Even though Weyuker presented the partial oracle in the context of testing non-testable programs, testers can apply an experience-based partial oracle to many kinds of programs. If a failure can be identified very efficiently using a partial oracle, it is not necessary to use or find a perfect oracle to find out the exactly correct outcome. Weyuker used the term pseudo oracle to refer to an independently written program that satisfies the same specification as the tested program (Weyuker 1982). The third approach to testing non-testable programs was using simplified data, which, in many cases, allows testers to verify the simple cases, for which the correct result can be determined, and extrapolate from this that the program works also for complicated data (Weyuker 1982).

It seems that, in most manual testing, the oracle problem is highly relevant and typically solved based on the experience-based knowledge of testers and varying types of documentation. Actually, in industrial practice, even in many test automation approaches, the oracle problem is left a human decision (Baresi & Young 2001) as, e.g., Whittaker (2000) wrote: "… such difficulty is the reason why the actual-versus-expected output comparison is usually performed by a human oracle: a tester who visually monitors screen output and painstakingly analyzes output data." From the empirical research on real-world testing activities, it is clear that an experience-based human oracle is, in many cases, the way in which tests are evaluated in practice; see, e.g., the ethnographic descriptions of Martin et al. (2007) and Rooksby et al. (2009). In the ET context, experience-based oracles have been presented, e.g., the heuristics-based approach. Kaner et al. (2002) and Bolton (2005) described the consistency heuristics, a set of rules for checking the consistency of functionality against various targets, such as the history of the product, comparable products, and users' expectations.

Some additional insight into the role of knowledge in defect detection is provided by the findings regarding the effects of domain knowledge on defect detection in the usability inspection and spreadsheet error-finding contexts. As described earlier, multiple studies on software testing have reported findings on the high importance of domain knowledge in testing (Beer & Ramler 2008, Iivonen et al. 2010, Kettunen et al. 2010, Merkel & Kanij 2010). Similar results have been reported in the context of usability testing. Følstad (2007) studied work-domain experts' performance as usability

evaluators and found that the findings of work-domain experts were classified as more severe and that the developers gave higher priority to items identified by work-domain experts (Følstad 2007).

To my knowledge, the oracle problem has not been studied in the context of manual testing with the intent of understanding, describing, or improving the way in which humans recognize defects. Some insight into the issue can be found in the area of end-user software engineering (Burnett et al. 2004). In the spreadsheet context, the oracle problem has been acknowledged and studied to some extent. First, Galletta et al. (1996) presented a conceptual model of the potential factors affecting error-finding performance. Their model classified the factors into four categories: individual factors, presentation factors, error factors, and external factors. They studied the individual experience factors (Galletta et al. 1993) and presentation factors (Galletta et al. 1996). They compared the error-finding performance of domain area (accounting) experts vs. novices and spreadsheet (software) experts vs. novices in finding domain-related and spreadsheet-related defects. They found that each type of expertise increased the error finding performance, but the performance of those with both types of expertise far exceeded the performance of other groups. The spreadsheet expertise increased the speed of revealing spreadsheet-related defects (Galletta et al. 1993). There is no research on how similar aspects, as presented by the conceptual model of Galletta et al. (1996), would affect defect detection performance in manual software testing.

Another important viewpoint regarding experience-based human oracles is the possibility of oracle mistakes. Testers do not always recognize a defect even if a test case reveals it; i.e., they make oracle mistakes. E.g., in the experiment of Basili and Selby (1987), the subjects recognized only 70% of observable failures, and Ruthruff et al. (2005) and Phalgune et al. (2005) studied oracle mistakes that end-user programmers made and found mistake rates ranging from 6% to over 20%. Oracle mistakes, meaning that a tester judges incorrect behavior to be correct or vice versa, are an important factor affecting the effectiveness and applicability of exploratory testing.

As a summary of the research on the role of experience and knowledge in testing, it can be stated that the importance of experience and, in particular, domain knowledge is recognized in the literature. There is no detailed understanding of the knowledge that is applied in testing. Existing studies in the software testing context are based on interviews regarding case studies and surveys, and the results do not provide insight into how testers actually work and apply their knowledge. The findings of the effect of experience and knowledge in defect detection in the context of usability evaluations and end-user programming support the importance of experience and do-

main knowledge. How and what type of knowledge is applied in defect detection when performing ET remains an unstudied area. This thesis aims at studying the ET activities and role of knowledge in defect detection activity of ET in detail.

3. Research goals and methodology

This thesis is an exploratory study of exploratory software testing. In this section, the goals of the research and more detailed research questions are described. As the main goals of this thesis are to increase understanding of the previously rather unstudied exploratory testing approach, the work is exploratory and theory-generating in nature (Patton 2002). Exploratory research is a proper research approach when researchers have little existing scientific knowledge about the activity under study but have a reason to believe that it is a relevant target of research (Stebbins 2001). To properly study these human aspects of software engineering, qualitative inquiry (Patton 2002) is used as the primary research method, combined with quantitative methods when necessary. This mixed-methods research approach that forms the overarching methodology for this thesis is described is Section 3.2. The sufficient details of each employed research method are described in the subsequent subsections.

3.1 Research goals and questions

The high-level research problem that this thesis aims to address is: *How do exploratory software testing approaches work in practice?*

To provide new understanding regarding the mostly unstudied ET approach, three goals are set for this thesis, and research questions are stated for each of the goals.

Goal 1: Define ET and understand the applicability of ET based on the literature.

RQ 1: How is ET defined in the literature?
RQ 2: How is ET motivated in the literature and in what contexts is ET claimed to be applicable?

The first goal of this thesis focuses on the existing knowledge of ET based on scientific and practitioner literature. The goal is descriptive and aims to

define the concept of ET and summarize the motivations and claims concerning ET that are presented in the literature. The goal is not to come up with a synthesis of results because the research results on ET are virtually non-existent.

Goal 2: Empirically investigate the benefits and shortcomings of ET.

RQ 3: What are the perceived benefits and shortcomings of ET in industry?

RQ 4: What is the defect detection effectiveness of the ET approach in comparison to the TCBT approach?

The second goal of this thesis focuses on producing empirical data regarding the benefits and shortcomings of the ET approach. The goal is twofold: first, to investigate the perceptions of industry practitioners who apply ET in their organizations and, second, to empirically compare the effectiveness of the ET and TCBT approaches.

Goal 3: Provide empirically based results on how the ET approach is applied in practice.

RQ 5: How is the ET approach applied in industry?

RQ 6: How is knowledge applied to failure detection in exploratory testing in industry?

RQ 7:How do people in different organizational roles contribute to defect detection?

The third goal of this thesis focuses on empirical results with respect to how ET is applied in industry. The goal is to provide descriptive results indicating the ways in which ET is applied as a part of software development activities. In addition, because the important role of knowledge in software testing is identified in the literature, this goal includes a detailed investigation on the knowledge aspect. Finally, directly related to the role of knowledge in testing is the question of who performs the actual defect detection activities, i.e., testing, in development organizations.

The results for each of the three research questions are covered in one or more articles that this thesis consists of. The mapping of the research goals and questions to the articles is presented in Table 1.

Table 1. Mapping the research questions and publications. "X" denotes that the article addresses the research question, and "**M**" denotes that the research question is a main research question in the article.

	I	II	III	IV	V	VI
Goal 1: Definition and applicability						
RQ 1: How is the ET defined in the literature?		**M**				
RQ 2: How is ET motivated in the literature and in what contexts is ET claimed to be applicable?	X	X				
Goal 2: Benefits and shortcomings						
RQ 3: What are the perceived benefits and shortcomings of ET in industry?		**M**				
RQ 4: What is the defect detection effectiveness of the ET approach in comparison to the TCBT approach?		X	**M**			
Goal 3: Application in practice						
RQ 5: How is the ET approach applied in industry?		**M**		**M**	X	
RQ 7: How is knowledge applied to failure detection in exploratory testing in industry?				X	**M**	
RQ 6: How do people in different organizational roles contribute to defect detection?						**M**

3.2 A mixed-methods approach

The overall research approach in this thesis is a mixed-methods approach (Creswell et al. 2003, Shull et al. 2008) that combines three main research approaches: case studies, experiment, and empirical observation-based field studies. Even though one of the studies is a controlled experiment, this research is mainly exploratory and hypothesis-generating in nature rather than existing hypothesis-verifying. This was due to the shallow body of literature and research on ET and, thus, the lack of existing theories or hypotheses. The primary research approach in this thesis is qualitative; however, a mixed-methods approach enables us to study this rather unstudied exploratory software testing phenomenon form diverse viewpoints. The research methods are mixed in this research both sequentially between studies and concurrently within a single case study (see Figure 1).

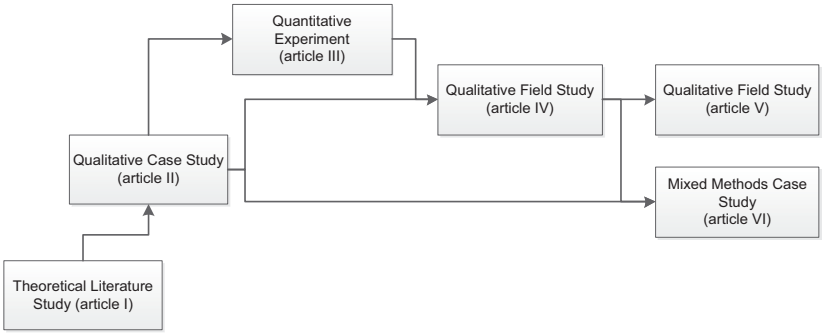


Figure 1. Mixed-methods approach and the studies of this thesis

In the first phases of this study, the sequential exploratory strategy for mixed-methods research (Creswell et al. 2003) was used to study the ET phenomenon. As Creswell et al. described, the primary focus in the sequential exploratory strategy is on exploring a phenomenon. In this research strategy, a primary qualitative study is conducted and a quantitative study is then applied to assist in the interpretation of qualitative findings or the test elements of an emergent theory resulting from qualitative findings (Creswell et al. 2003). The first qualitative case study (article II) was performed to understand the role, benefits, applicability, and shortcomings of exploratory testing in software development organizations. Based on the results of the first case study and a literature review, hypotheses of the defect detection effectiveness of ET in comparison to TCBT emerged. These hypotheses were tested in a controlled experiment (article III). In the first case study, some quantitative data was also used to enrich the qualitative descriptions, as in the concurrent nested strategy (Creswell et al. 2003).

The results of the experiment were not conclusive and required more explanation. Because ET is a practice that has not been studied much and the descriptions in the books and practitioner reports are somewhat vague, more qualitative research was needed. The study continued, following a sequential explanatory strategy (Creswell et al. 2003) in which subsequent qualitative studies were conducted to gain more insight into the ET phenomenon, focusing on the defect detection activity. Two empirical observation-based field studies (articles IV and V) were conducted to study the actual exploratory testing practices of software development professionals.

During the research, it was recognized that, in practice, the knowledge of professionals in a variety of organizational roles seemed to be an important factor in testing and defect discovery. Another exploratory case study (article VI) was performed to better understand the contributions of different employee groups in defect detection. In this last study, the concurrent triangulation strategy (Creswell et al. 2003) was used by mixing quantitative and qualitative methods within the case study. Qualitative and quantitative methods were used partly to answer different questions and to triangulate and utilize multiple sources of evidence, but qualitative data was also employed in explaining the quantitative results of the study.

In the next three subsections, the case study, experimental, and observation-based field study methods used are introduced.

3.3 Literature reviews

A literature review was a part of all of the studies of this thesis. Article I was based purely on a review and theoretical analysis of literature. In articles II, III, and V, a more comprehensive review of ET literature from the viewpoint of the research questions of each article was performed.

The literature reviews did not follow any specific systematic literature review (SLR) protocol. Instead, the literature reviews used an informal review process that covered academic databases, practitioner literature, and Internet sources. The literature reviews were carried out by searching for relevant material from the following sources:

- IEEE Explorer database
- ACM Digital Library database
- Scopus database
- Google Scholar
- Generic Internet search services: Google and Yahoo!

All references in each relevant article found were reviewed. In addition, a large number of known software testing textbooks were reviewed. Formal inclusion and exclusion criteria were not applied since it was feasible to include all published material because no reliable research was available. In practice, articles, books, and reports were included and other material such as Web pages and slide presentations were excluded.

The reasons for using an informal approach instead of an SLR were as follows. Performing an SLR on exploratory testing was not feasible since there is no established body of research on exploratory testing, which means that systematic searches in academic databases with such relevant keywords as "exploratory software testing" did not return any relevant findings, and more generic search terms resulted only in the retrieval of thousands of irrelevant papers. More importantly, the relevant papers did not necessarily include the used search terms. Finally, the goal of the literature review was not to draw conclusions based on a synthesis of existing research but, rather, to summarize the existing published knowledge, which did not justify the effort and rigor of the SLR approach.

3.4 Case studies

The two case studies in this thesis are both industrial multiple case studies with three software product development organizations as the units of analysis (articles II and VI). In the first case study, the methodology was designed mainly according to Yin (1994) and, in the second case study, it was designed according to Runeson and Höst (2009).

The selection of case organizations in the case studies was based on availability through ongoing research cooperation and the representativeness of the case organizations in terms of the studied phenomena. In the first case study, case organizations were selected that applied an exploratory testing approach in their software development. In the second case study, the availability of quantitative defect data was an additional selection criterion.

The first case study (article II) was a descriptive case study where the applied ET approaches in three case organizations were studied using semi-structured interviews and qualitative data analysis as the primary research methods. The number of interviewees was 1–4 in each case organization. The qualitative descriptions were enriched by quantitative data regarding defect detection effectiveness and efficiency.

The second case study (article VI) was an exploratory case study where the role and contributions of different organizational groups in defect detection were studied. The goal of the study was to provide initial answers to the research questions and to generate hypotheses for future studies. The case study was conducted with three case organizations as the units of analysis. In this study, multiple methods were applied, combining qualitative and quantitative data and analysis. Qualitative data was collected using semi-structured interviews with four interviewees from each organization, collaborative workshops, and informal communication (discussions and email communication) as the methods. Qualitative data analysis was performed by, first, transcribing the interview recordings and, second, performing qualitative coding using both preformed codes and open coding followed by axial coding rounds (Strauss & Corbin 1998). In addition to the qualitative methods, an in-depth quantitative analysis of the data on reported defects in the defect databases in each case organization was performed. After the quantitative defect data analysis, additional validating interviews were conducted to validate and gain further explanations for the generated hypotheses.

3.5 Controlled experiment

In the third study of this thesis (article III), a controlled student experiment was conducted. The study focused on comparing the defect detection effectiveness of the test case-based and freestyle exploratory testing approaches.

3.5.1 Overview

A one-factor block design with a single blocking variable (Juristo & Moreno 2001) was used as the design of this experiment. The empirical research guidelines presented by Kitchenham et al. (2002) were used as a guide in designing this experiment. The study was performed as a student experiment in the context of an undergraduate-level software testing course. The subjects were randomly divided into two groups, both of which performed similar test sessions with and without predesigned test cases. The ordering of the test sessions differed for the two groups. The test sessions were controlled sessions in which 90 minutes of effective testing time was given for testing the features. The subjects of both groups performed the sessions at the same time in different rooms. In both sessions, the same application was tested, but the feature set being tested was different in the first and second sessions.

The experiment consisted of three separate phases: preparation, session 1, and session 2. In the preparation phase, each subject designed and documented test cases for the feature set that was allocated for TCBT for them. The subjects designed the test cases without supervision and used as much effort as they required for the preparation phase. Note that each student designed the test cases only for TCBT; they did not prepare test cases for the other feature set that was tested using an exploratory approach. All subjects participated in both sessions, but the ordering of the test approaches was different for the two groups. The structure and length of both controlled testing sessions were exactly the same. An overview of the experimental arrangements is described in Table 2.

Table 2. Experiment arrangements

	Group 1	**Group 2**
Preparation	Test cases for feature set A	Test cases for feature set B
Testing Session 1	TCBT Feature set A	ET Feature set A
Testing Session 2	ET Feature set B	TCBT Feature set B

3.5.2 Factors and blocking variables

The factor in this experiment is the applied testing approach. The factor has two alternatives: test case based testing (TCBT) and exploratory testing (ET).

Blocking variables represent the undesired variations in the experimental design that cannot be eliminated or made constant. In this experiment, the significant blocking variable was the tested feature set. The feature set also

included the seeded and actual defects that, consequently, were not the same for all of the elementary experiments. The type of tested features and the properties of the defects in the tested software variant have an effect on the test results. For these reasons, the tested feature set was considered a blocking factor in the experimental design, and we took into account the possible effects of two different feature sets in the data analysis.

3.5.3 Response variables

This study looked at the defect detection effectiveness measured by the number of defects found during a fixed-length testing session. In addition, insight into the effectiveness was gained by analyzing the proportions of different defect types and severities. In addition, the number of false defect reports produced during a testing session was used as one variable.

3.5.4 Subjects

The number of subjects in the experiment was 79. The major undesired variation that affected this experiment originated from the individual differences in the student subjects. These properties included experience in software engineering, amount of study, prior training in software testing, and individual skills. These variations were handled by two means. First, all subjects performed the experiment two times, once using each of the testing approaches. Second, the subjects were randomly assigned into two groups that applied the two approaches in different orders. The two groups were used for the sole purpose of randomizing the application order of the two approaches, and the testing assignments in this experiment were individual tasks for each subject.

3.5.5 Data collection and analysis

The experiment data was collected in many forms. First, subjects submitted the predesigned test cases in an electronic format. Second, in the testing sessions, the subjects filled in test logs and defect report forms. Third, after each session, the subjects filled in a survey questionnaire.

The defect report data was analyzed in detail. Each found defect was recorded with all details and imported into the SPSS statistical analysis tool, which was used for all statistical analysis. The number of defects detected by the ET and TCBT groups were compared using the t-test. In addition, a multi-factorial analysis of variance (ANOVA) was used to control for and understand the effect of the different feature sets and the possible interactions between the feature set and the testing approach.

To analyze the defect types and number of false reports, the defect distributions of the ET and TCBT groups were presented and a significance analysis using the non-parametric Mann-Whitney test was performed.

3.6 Observation-based field studies

Two of the studies of this thesis (articles IV and V) are field studies that focus on the testers' activities in real testing work and in authentic environments and contexts, i.e., studying testing activities and defect detection in situ. In these studies, field observations were used as the data collection method, and the data analysis was performed primarily using qualitative analysis and an applied grounded theory approach. The observations were augmented with semi-structured interviews to get additional insights.

3.6.1 Selection of organizations and observation sessions

The studied software development organizations for these two field studies were selected based on the use of the ET approach and accessibility through existing research collaboration.

The subjects for the observations were selected based on the recommendations of the test or development managers in each company. In the second observation study (article V), the selection criterion was to find high-performing testers in terms of the subjective opinion of the managers. In the selection of the individual test sessions, the goal was to find functional testing activities that tested features through human-useable interfaces and included new functionality or major changes. The selection of observed sessions was affected by the testers' and researcher's schedules and the availability of suitable testing activities.

3.6.2 Observation method

In both field studies, the research method was participant observation (Seaman 1999). Using observations as a method gives access to the actual testing tasks that the subjects perform in their authentic working environments. It is common for practitioners to have difficulty describing, e.g. in interviews, the actual work activities and how they perform their work in practice. With direct observation, "the inquirer has the opportunity to see things that may routinely escape awareness among the people in the setting" (Patton 2002). When observations are used, the researchers are not relying on descriptions and conceptualizations by the subjects based upon their own recollection of how they perform their work.

Participant observations in this thesis are used based on the definition of Seaman (1999), who stated that the idea of participant observations is to "capture firsthand behaviors and interactions that might not be noticed otherwise" and that "participant observation does not necessarily imply that the observer is engaged in the activity being observed" (Seaman 1999). However, since the definition of participant observation is not consistent among sources (see, e.g., Seaman & Basili 1998, Lethbridge et al. 2005, Patton 2002), I summarize the method used in this thesis in Table 3, applying the six dimensions of fieldwork variations presented by Patton (2002).

Table 3. Properties of the participant observation method

Role of the observer	Onlooker. The observer sat beside the subject for the entire testing session. The observer did not carry out or participate in any way in the actual testing activities.
The perspective of the observer	Outsider dominant. The observer was not a part of the organization or involved in the product development. The observer was, however, familiar with the organization, the tested software products, and to some of the subjects through longer research cooperation with the organizations.
Number of observers	Single researcher. The author of this thesis.
Disclosure of the observer	Fully disclosed to the subjects. The observer was clearly present in the testing situation, and the observed subject was strongly conscious of his presence. Even though the observer tried to be as inconspicuous as possible, the subjects communicated directly to the observer during the observations.
Duration of the observation	One or two observation sessions per subject. The lengths of the observation sessions varied between 1-2.5 hours.
Focus of the observations	Narrowed to the individual test execution tasks of single testers. Any test planning, design, documentation, and management activities outside the observed testing sessions were excluded.

In exploratory testing work, much of the interesting behavior happens inside a tester's head. To get data on the mental processes that take place during testing sessions, a method is needed to record what the observed tester was doing and thinking during the testing. A commonly used method for this purpose is the think-aloud protocol approach (Patton 2002, Hughes & Parkes 2003). The think-aloud method was applied in the observations by asking the subjects to think aloud, i.e., describe what they were doing and thinking during the testing session. The goal of these studies was to observe testing sessions that were as authentic and natural as possible and, thus, the researcher did not enforce continuous verbalization but only briefly encouraged the subject to verbalize every now and then. The goal of using the think-aloud method was not to perform direct verbal protocol analysis (Hughes & Parkes 2003) but, instead, use the subject's verbalizations in the analysis together and as a part of the video-recorded data.

The context of the observations was software development professionals performing their actual testing tasks in their normal working environment. Most of the sessions took place in front of the subjects' personal work desk

using a personal computer. The total number of observed sessions in the first and second field studies was 11 and 12, respectively.

3.6.3 Data collection methods

The data collection methods differed between the first and second field studies. The first field study relied purely on written field notes that were produced during the observation sessions and short interviews and memos done right after the sessions. The field notes were recorded using a pre-planned structure in which four main categories were identified: session data, test ideas, found defects, and general notes. Under each category except general notes, a varying number of details were planned to ensure that all aspects of the observed behavior that were deemed relevant beforehand were rigorously recorded.

In the second field study, the data recording was carried out by comprehensive video- and audio-recording of all observation sessions, augmented with written field notes. Video-based field observations have been used as a research method in some software engineering studies (Höfer 2008, Salinger et al. 2008, Wu et al. 2009). The observation sessions were recorded using two cameras. The field notes were recorded in written format using laptop computer.

In both field studies, the test documentation that was used during the observed sessions as well as all defect reports were collected to support the analysis. In addition to observations, short interviews were conducted after each observation session. The interviews covered the background information of the subject and discussion on how typical, for the subject, the observed sessions were overall and in terms of the detected defects and issues. In the interviews, we used a general interview guide approach (Patton 2002).

3.6.4 Data analysis methods

In the first field study, qualitative data was collected as structured field notes by during the observation sessions. The field note data was analyzed by coding the findings and identifying categories. A pre-defined initial coding scheme was used first to code the data using high-level codes based on the research questions. The preliminary list of codes was refined and extended during the analysis work, as described in (Miles & Huberman 1994). After the first coding round, the concepts describing the testers' practices were identified as categories, and coding was repeated to improve and verify the findings. Clustering (Miles & Huberman 1994) the findings based on the purpose coding led to five partly overlapping clusters.

The goal of the second field study was to gain understanding of how defects are identified by testers performing exploratory software testing and the role of personal knowledge in it. Grounded theory (GT) (Strauss & Corbin 1998) and, more specifically, so-called "Straussian" (van Niekerk & Roode 2009) grounded theory was selected as the general research approach for qualitative analysis in this study. In GT, the analysis is grounded to the data instead of existing theories, and the research is theory-generating. However, the rich video data, as primary documents, poses certain challenges in applying the pure GT approach, which has been reported also by other researchers (Salinger et al. 2008). The amount of detail and all the potentially relevant nuances of the recordings combined with the required effort and time consumption associated with the coding directly to the video material proved to be too difficult.

To overcome these challenges, we applied three modifications to the GT approach. First, we selected the perspective of our analysis before the coding phase to focus the analysis on the relevant issues. Second, we preselected samples of the data, based on the chosen perspective, to limit our analysis. Third, we transcribed the video data before the actual coding, which was applied to the transcribed text.

We performed the data analysis in four phases. First, we performed open coding with the perspective of the testers' activities directly to the full-length observation video-recordings. Second, based on the research questions, all excerpts from the video-recordings that were associated with failure detection were selected. Third, because of the aforementioned challenges of coding directly to video episodes, the selected episodes were transcribed in written format. In these transcriptions, not only the think-aloud protocol but also the behavior of the tester, the general approach of testing, the context, and the observed symptoms of the detected failure itself were described. Fourth, by using these detailed transcriptions, the open coding of the defect detection episodes continued. In this phase, the open coding and axial coding were intertwined cyclic activity. As new codes emerged, they were compared and grouped with similar codes, and categories were identified around groups of codes describing similar concepts. When the categories and classes emerged in the analysis, the transcriptions were analyzed again against those concepts in a cyclic manner to confirm the findings.

4. Summary of the results

In this section, the results of the research are summarized. The details of the results and each individual study are found in the respective articles. This section is structured in accordance with the three research goals introduced in Section 3.1. Under each research goal, the results of the corresponding studies are presented in relation to each of the research questions.

4.1 Goal 1: Define ET and understand its applicability

The first goal of this thesis addresses the definition and applicability of the exploratory testing approach based on literature studies. The research results are summarized for both the definitions and applicability of the ET approach in the subsequent subsections.

4.1.1 Definitions of the ET approach in the literature

A literature review of the exploratory testing literature was performed and presented in the related work of article II. The findings of the literature review summarized the definition of ET.

As a synthesis of different definitions and descriptions of ET in the literature, the following properties that characterize the exploratory approach to testing were presented:

1) Tests are not defined in advance as detailed test scripts or test cases. Instead, exploratory testing is exploration with a general mission without specific step-by-step instructions on how to accomplish the mission.
2) Exploratory testing is guided by the results of previously performed tests and the knowledge gained from them. An exploratory tester uses any available information about the target of testing, such as a requirements document, a user's manual, or even a marketing brochure.
3) The focus in exploratory testing is on finding defects by exploration instead of systematically producing a comprehensive set of test cases for later use.

4) Exploratory testing is simultaneous learning of the system under test, test design, and test execution.
5) The effectiveness of the testing relies on the tester's knowledge, skills, and experience.

4.1.2 Motivation and applicability of the ET approach

The literature review of article II covered the applicability of ET and the claimed benefits and shortcomings of ET.

Applicability

The claims and propositions of the applicability of ET in the literature were analyzed and the following contexts were identified in which ET is proposed to be a highly applicable testing approach:

- There is not enough time for systematic testing approaches.
    - Rapid feedback or learning of the product is needed.
- ET should be planned as a part of the testing approach in most software development projects.
    - ET can be used to provide more diversity for scripted tests.
    - ET is also useful when test scripts become "tired"; i.e., they are not detecting many defects anymore.
- ET fits well into testing from an end-user viewpoint.
- ET is a good way to investigate the status of particular risks.
- Regression testing based on defect reports can be done by exploring.
- Situations in which choosing the next test case to run cannot be determined in advance but must be based on previous tests and results.
- ET can be used to explore the size, scope, and variations of a found defect to provide better feedback to developers.

Benefits

In the literature review, five claimed benefits of the ET approach were identified. The most commonly claimed benefit of ET is the effectiveness of testing in terms of the number and importance of found defects. Claims were made that, in some situations, ET can be orders of magnitude more efficient than scripted testing. A second benefit of exploratory testing is simultaneous learning. When testers are not following pre-specified scripts, they are actively learning about the system under test and gaining knowledge about the behavior and the failures in the system. This is claimed to help testers come up with better and more powerful tests as testing proceeds. A third benefit is the ability to minimize preparation documentation before execut-

ing testing. This is an advantage in a situation where the requirements and design of the system change rapidly or in the early stage of product development when some parts of the system have been implemented but the probability for major changes is still high. A fourth benefit is the ability to perform exploratory testing without comprehensive requirements or specification documentation because exploratory testers can easily utilize all the experience and knowledge of the product gained from various other sources. Finally, the rapid flow of feedback from testing to both developers and testers is a benefit of ET. This feedback loop is especially fast because exploratory testers can react quickly to changes to the product and provide test results back to developers.

Shortcomings

Identified shortcomings of ET in the literature were rare. Only two negative aspects of exploratory testing were identified. One is the difficulty of tracking the progress of individual testers and the testing work as a whole. It is considered difficult to find out how the work proceeds, e.g., the feature coverage of testing, because there is no planned low-level structure that can be used to track the progress. The other shortcoming that was pointed out is the fact that ET has no ability to prevent defects. Designing the test cases in scripted testing can begin during the requirements-gathering and design phases and, thus, reveal defects early.

Analysis of quality practices of agile development methods

This thesis research was initiated in article I, in which the testing and quality assurance approach of agile software development methods was analyzed. The agile principles were analyzed form the viewpoint of software testing, and the challenges that agile principles pose for testing were identified as presented in Table 4.

Table 4. Challenges that agile principles pose for testing (article I)

Agile principle	Challenge
Frequent deliveries of valuable software	- Short time for testing in each cycle - Testing cannot exceed the deadline
Responding to change even late in the development	- Testing cannot be based on completed specifications
Relying on face-to-face communication	- Getting developers and business people actively involved in testing
Working software is the primary measure of progress	- Quality information is required early and frequently throughout development
Simplicity is essential	- Testing practices easily get dropped for simplicity's sake

From the viewpoint of traditional principles of software testing, the agile testing practices were analyzed to identify contradictions (see Table 5) that might represent potential quality assurance challenges in agile methods.

Table 5. Traditional testing principles and contradictory practices in agile methods (article I)

Testing principle	Contradictory practices in agile methods
Independence of testing	- Developers write tests for their own code - The tester is one of the developers on a rotating role in the development team
Testing requires specific skills	- Developers do the testing as part of the development - The customer has an important and collaborative role and a lot of responsibility for the resulting quality
Oracle problem	- Relying on automated tests to reveal defects
Destructive attitude	- Developers concentrate on constructive QA practices, i.e., building quality into the product and showing that features work
Evaluating achieved quality	- Confidence in quality through tracking conformity with a set of good practices

To deepen the analysis of the general quality assurance challenges of agile development, the quality practices of four specific agile development methodologies were analyzed using the Cycles of control framework of time-paced software development (Rautiainen 2004). Based on the two theoretical analyses, the shortcomings of quality assurance in agile methodologies were identified, and session-based exploratory testing approach along with an independent tester role for the heartbeat time horizon were proposed as improvements to agile practices.

4.1.3 Key findings

Based on literature review, a definition of the exploratory software testing approach was synthetized and its applicability, claimed benefits, and shortcomings described. The applicability of ET in the agile software development context was shown through a theoretical analysis of the quality assurance and testing practices of agile development methods. The contribution of this analysis (article I) for the goals of this thesis is motivational. The results of the article give a concrete description of a context in which the exploratory testing approach is applicable. It shows the gaps in the quality practices of agile software development and concludes that the exploratory testing approach matches agile development practices and principles and could serve as an improvement in terms of the described gaps. The hypothesis of exploratory testing as a vehicle for utilizing domain knowledge in testing was raised in article I.

4.2 Goal 2: Investigate the benefits and shortcomings of ET

The second goal of this thesis was to study the benefits and shortcomings of ET using empirical research methods. This goal was covered in two of the studies of this thesis. First, a multiple case study (article II) was conducted to determine the benefits and shortcomings of ET as perceived by practitioners in industry. Second, a controlled experiment (article III) was performed in a student context to study the defect detection effectiveness of ET in comparison to test case-based testing (TCBT). The results of these two studies are summarized in the next two subsections.

Table 6. Reported reasons for using ET in the three cases (article II)

Reasons for using ET	1	2	3
The software can be used in so many ways or there are so many combinations between different features that writing detailed test cases for everything is difficult, laborious, and even impossible.	X	X	X
It suits well to testing from a user's viewpoint.	X	X	X
It emphasizes utilizing the testers' experience and creativity to find defects.	X	X	X
It helps provide quick feedback on new features from testers to developers.	X	X	X
It adapts well to situations, where the requirements and the tested features change often, and the specifications are vague or incomplete.	X	X	
It is a way of learning about the system, the results of which can be utilized in other tasks, such as customer support and training.		X	X

4.2.1 Perceived benefits and shortcomings of ET in industry

In a multiple case study, the perceived benefits and shortcomings were studied in three software development organizations. In addition, the organizations' motivations for using ET were described.

Table 7. Perceived benefits of the ET approach (article II)

Benefit	Description
Versatility	ET is more versatile and goes deeper into the tested feature. Testers test things that they would not include in test cases. Examples of such tests include testing the dependencies of new and existing features based on expertise and knowledge of the system. Another example of versatility is retesting a fixed defect, where testing is not restricted to just retest in the same way as before, but includes exploring for possible new defects at the same time.
Effectiveness and efficiency	ET helped to find important defects in a short amount of time, but if a less experienced person with less domain knowledge would do the testing, the results might not be so good. More defects were found in system testing using ET than using TCBT, because the test cases are designed to verify that the system works and the testers use ET with a more destructive attitude. Using ET to test features of a complex system can be very time consuming.
Better overall view of quality	Getting an overall picture of the quality of the system quickly is one aspect of efficiency. This was important because the information gained from ET was used as a basis for prioritizing the work towards the end of the project.

The results concerning the reasons for using the ET approach are summarized in Table 6. In the table, the reasons that interviewees at each of the three organizations gave for using the ET approach are listed. These reasons are mostly related to the perceived benefits of ET as well as to the shortcomings or challenges of applying the TCBT approach. The additional benefits of ET are described in Table 7. The benefits shown in this table were not used as reasons for applying the ET approach but were described by the interviewees when they were questioned further about the benefits of ET.

The perceived shortcomings identified in the case study are summarized in Table 8.

Table 8. Perceived shortcomings of the ET approach (article II)

Shortcoming	Description
Test coverage	Coverage in one form or another was the biggest shortcoming of ET. Challenges concerning coverage included planning and selecting what to test, since the limited time and resources restricted the amount of testing. It was a question of prioritizing testing to potential weak spots in the system and trying to allocate time of domain experts for testing.
Test tracking	Lack of control of the test coverage combined with scarce documentation of the testing itself created a challenge of following up what had been tested and what had not.
Dependency of individual testers	Relying on the expertise of the testers made ET more prone to human errors than TCBT. It was impossible to find testers with enough experience to act as professional users. Another challenge was that all testers have different backgrounds and experience and thus perform ET from different viewpoints. This was seen both as a strength and weakness, especially regarding the versatility of testing this implied.
Repeatability	The repeatability of defects was seen as a shortcoming of ET at on case organization. This was related to a complex system that permitted many ways of performing tasks, and each task could require up to a hundred or more steps.

4.2.2 Defect detection effectiveness and efficiency of ET

In the case study (article II), quantitative data on the number of detected defects and the testing effort were collected. In two cases where a session-based testing approach was used, the average numbers of detected defects per testing hour were 4.8 and 8.7. These findings support the hypothesis regarding the effectiveness and efficiency of the ET approach, but the result is not conclusive due to the severe limitations of this data.

In addition to the case study, another study was conducted in which a controlled experiment methodology was used (article III). The fundamental difference between ET and TCBT is that ET does not rely on pre-designed and documented test cases. Thus, a controlled student experiment was car-

ried out. The experiment compared the freestyle ET approach to the TCBT approach in terms of defect detection effectiveness. In the experiment, 79 advanced software engineering students performed manual functional testing on an open-source application, JEdit text editor, with actual and seeded defects. Each student participated in two 90-minute controlled sessions, using ET in one and TCBT in the other. The study focused on *identifying the effect of using predesigned and documented test cases in manual functional testing with respect to defect detection performance.*

This study looked at the defect detection effectiveness measured by the number of defects found during a fixed-length testing session. Additionally, more insight into the efficiency is gained by considering the proportions of different defect types and severities as well as the number of false defect reports produced. In this section, a summary of the results of the experiment is presented based on the statistical analysis of the data.

Defect counts

The main response variable in the experiment was the number of detected defects in a fixed-length testing session. The defect count data is summarized in Table 9 and Figure 2. The number of defects found in each feature set is different due the differences in the feature sets used. The absolute mean defect counts for the ET and TCBT approaches were 7.038 and 6.367 respectively, the difference showing 0.671 more defects in the ET approach. The difference, however, was not statistically significant. There was no difference in the number of detected seeded defects between the approaches, but the ET approach detected more real (non-seeded) defects.

Even though the length of the testing sessions was fixed at 1.5 hours, it should be taken into account that, in the TCBT approach, the subjects spent, on average, 7 hours designing the test cases, which means that the TCBT approach took 8.5 hours of total effort, i.e., almost six times more than ET, to reach the same results.

Table 9. Summary of the defect count data (article III)

Testing approach	Feature set	Number of defects	Found defects per subject	
			mean	Std. dev.
ET	A	44	6.275	2.172
	B	41	7.821	2.522
	Total	85	7.038	2.462
TCBT	A	43	5.359	2.288
	B	39	7.350	2.225
	Total	82	6.367	2.456

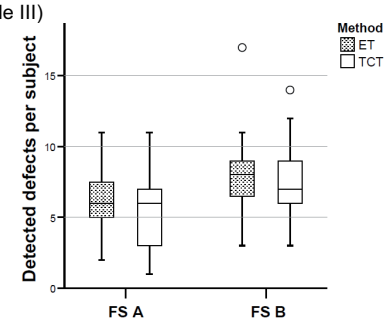


Figure 2. Defect counts (article III)

Types of defects

The results of the analysis of defect type and severity were used to provide a deeper understanding of the differences between the two testing approaches. Table 10 characterizes the defects based on the detection difficulty.

Table 10. Detection difficulty distribution (article III)

Mode	ET	TCBT	ET/TCBT	Total
0 = easiest	120	93	129 %	213
1	327	320	102 %	647
2	89	75	119 %	164
3 = hardest	20	15	133 %	35
Total	556	503	111 %	1059

A mode 0 defect means that the defect is immediately obvious to the tester, e.g., a missing button. A mode 1 (1-way) defect requires one action of the tester to cause a failure and reveal the defect, e.g., save a file to find out that some part of the file is not saved. Double-mode (2-way) and triple-mode (3-way) defects require a combination of 2 and 3 actions or inputs, respectively, to make the failure occur. In Table 10, we can see that ET found more defects in all classes of detection difficulty. The most notable differences were for mode 0 and mode 3 defects, for which ET found 29% and 33% more defects than TCBT. However, the Mann-Whitney U test showed the differences to be statistically insignificant for all classes.

In Table 11, the numbers of detected defects are categorized based on their technical type. There were no radical differences in the number of defects with different technical types. ET found more wrong-function defects, GUI defects, and usability problems than TCBT and fewer technical defects. The differences in the documentation, inconsistency, and usability categories are unreliable due to the small absolute numbers of defects in these categories.

Table 11. Technical type distribution (article III)

Type	ET	TCBT	ET/TCBT	Total
Documentation	8	4	200 %	12
GUI	70	49	143 %	119
Inconsistency	5	3	167 %	8
Missing function	98	96	102 %	194
Performance	39	41	95 %	80
Technical defect	54	66	82 %	120
Usability	19	5	380 %	24
Wrong function	263	239	110 %	502
Total	556	503	111 %	1059

In Table 12, the defects are categorized based on their severity. ET found 64% more negligible defects, 32% more minor defects, and 14% more normal defects. TCBT found 5% more severe and 2% more critical defects.

Table 12. Severity distribution (article III)

Severity	ET	TCBT	ET/TCBT	Total
Negligible	23	14	164 %	37
Minor	98	74	132 %	172
Normal	231	203	114 %	434
Severe	153	160	96 %	313
Critical	51	52	98 %	103
Total	556	503	111 %	1059

The data for false defect reports, meaning defect reports that were incomprehensible, duplicated, or reported non-existent defects, are summarized in Table 13. TCBT produced, on average, 1.05 more false reports than ET.

The Mann-Whitney U test for statistical significance showed that the effect of the testing approach on the number of false defect reports was highly significant, with a two-tailed significance of 0.000.

Table 13. False defects (article III)

Testing approach	Feature set	False defects per subject	
		$\bar{x}$	σ
ET	A	1,00	1,396
	B	1,05	1,191
	Total	1,03	1,291
TCBT	A	1,64	1,564
	B	2,50	1,867
	Total	2,08	1,767

4.2.3 Key findings

The description of the motivations of ET use and the perceived benefits and shortcomings of ET in the companies were the main contributions of article II. The results of the study support many of the benefits claimed in the existing literature, including the hypothesis of high defect detection effectiveness and efficiency. In addition, the results reveal some new findings. First, the use of ET for learning the system for purposes other than better testing was not reported in the literature. In two of the three case companies, one of the reasons for using ET was to learn the features and behavior of the system, e.g. to help prepare training materials and answers for customer service purposes. Second, the potential shortcomings were identified regarding test coverage, strong dependence on the expertise and personal properties of individual testers, and repeatability of the defects. Based on

the case study results, the hypothesis regarding the importance of domain knowledge in ET was strengthened, and a need for future research was identified.

The results of article III made four contributions. First, a lack of research on manual testing activities from points of view other than the test case design point of view was identified. Second, the results showed no benefit, in terms of defect detection effectiveness, in using pre-designed test cases in comparison to a freestyle exploratory testing approach. Third, there appeared to be no significant differences in the detected defect types, severity, and detection difficulty. Fourth, the results indicate that TCBT produces more false defect reports than exploratory approach. Even though the null hypothesis could not be rejected, the results strengthen the hypothesis of ET as an effective and cost-efficient approach to functional software testing. The hypothesis regarding efficiency is particularly considerable if we take into account the significant effort used to pre-design test cases in the TCBT approach, which was avoided in the ET approach.

4.3 Goal 3: Study how ET is applied in practice

The research regarding the third goal, how the ET approach is applied in practice, was performed in two empirical observation studies (articles IV and V) and in two multiple case studies (articles II and VI). The more focused research questions under this goal were, first, how is the ET approach applied in industry; second, how is knowledge applied to defect identification in exploratory testing in industry; and third, how do people in different organizational roles contribute to defect detection?

The results in terms of each of the three research questions are covered in the following subsections.

4.3.1 ET approaches in industry

In the multiple case study on ET in three software product companies, a description of different ways of applying the ET approach was presented. The six different ET approaches are summarized in Table 14.

The session-based ET approach followed the description of Bach (2000). The session based approach was motivated by the focusing effect; most of the interviewed persons found it beneficial to isolate the testing time into focused sessions without other tasks or interruptions. The exploratory regression testing approach, as well as the smoke testing, to some extent, differs from the typical focus on automation and the high level of repeatability of regression testing in the literature. In these companies, regression testing

was not performed exhaustively over the whole system. Rather, it concentrated on the changes and fixes made and, based on the tester's experience, exploring possible new and related defects caused by the fixes. The main reasons mentioned for this kind of "limited" regression testing were lack of time or resources for complete regression testing of the system.

The important findings also included the outsourced exploratory testing for utilizing the domain knowledge of expert users and the frequent use of freestyle ET as a part of other duties.

Table 14. Exploratory testing approaches (article II)

ET Approach	Description
Session-based ET	Testing was organized in test sessions during which the tester accomplished one planned testing task without any interruptions or other disturbance. The sessions were planned using short descriptions that described briefly the testing task, goals of the test session, and the target of testing. There was no systematic higher level planning or control of the coverage of testing.
Functional testing of individual features	ET for testing individual features right after the feature was implemented. This was performed by persons from the requirements management team and focused on testing whether the implementation corresponded to the requirements and the designer's actual ideas of the specified functionality or not. This enabled fast feedback to the developers in the early phase of the development life cycle.
Smoke testing	Each of the releases was smoke tested by the service team. This exploratory testing took from half an hour to a day and was guided by a "heading-level" list of the areas to be tested. In addition, every fix and enhancement was checked to ensure that the reported fixes actually had been performed and worked as the service team member would expect from the end-user point of view.
Exploratory regression testing	Exploratory testing to verify fixes and changes after implementing a single fix. A tester took a short testing session to verify the fix, typically without any planning or formal tracking or control. The result of this session was informally communicated to the developer or, if it was a defect fix, through the defect tracking system.
Subcontracted exploratory testing	Real users of the system were used as subcontracted testers. Experienced professional users of the system were hired to test the upcoming release. This testing was organized by features and the task of the testers was to perform real working scenarios and explore each feature of the software.
Freestyle exploratory testing	Unmanaged exploratory testing as part of other duties. It was common to test the latest alpha and beta releases, for example, at customer services as a part of the everyday work. Exploratory testing was quite often used as part of systematic system testing to explore functionality beyond the documented test cases. The intent was to find more defects and defects that are not straightforward to find.

Another study on the actual exploratory testing practices in industry was based on field observations, in which 11 practitioners were observed while they performed their actual testing activities. As a result of this qualitative analysis, a classification of testing practices was created (see Table 15). In

this framework, the 22 identified practices were classified into 9 test session strategies and 13 detailed test execution techniques.

Table 15. Classification of testing practices (article IV)

Test session strategies	Exploratory	6 practices
	Documentation based	3 practices
Test execution techniques	Exploratory	6 practices
	Comparison	4 practices
	Input	3 practices

The important contributions of this study included the finding that many of these techniques that were applied in an exploratory way were actually based on theories and assumptions that are the same or similar to some of the traditional test-case design techniques. As an example, the techniques in the input technique category were similar to the classic equivalence class partitioning and boundary value analysis techniques (Myers 1979). The *covering input combinations* technique, on the other hand, captured the basic idea of the combinatorial testing. Many of the exploratory strategies and techniques were similar to the general heuristics, rules of thumb, and experience-based lessons found in software testing textbooks (Myers 1979, Kaner et al. 2002). The difference between the techniques observed and how the techniques are presented in the literature is that the execution-time practices were used as part of test execution, not as test design methods beforehand.

Another new finding was the identification of comparison techniques that are not often described in the testing literature because of the test case assumption, i.e., assuming that expected results are documented in the test cases and, thus, that the comparison is a non-issue. As a notable exception, Kaner et al. (2002) listed some evaluation-based techniques including comparison techniques and consistency heuristics.

As a conclusion, we state that this study provides the initial results of a research study on the manual exploratory testing practices in the context of how testing is practiced in industry. The classification of the testing practices helps in better understanding the numerous findings and supports future research. This study supports the hypothesis that testers, in practice, apply numerous techniques and strategies during test execution and do not mechanically rely on test documentation. On the other hand, testers clearly need testing techniques even when applying experience-based and exploratory testing approaches. Finally, we identified that execution-time techniques are partly similar to test-case design techniques but are strongly ex-

perience-based and applied in a non-systematic fashion during test execution.

4.3.2 Role of knowledge in failure detection

The question of how testers apply their knowledge in defect detection was studied with rigorous field observations and applied grounded theory analysis. This study is reported in article V. The goal of the research was to study how failures are identified during actual exploratory testing work. As a result of the analysis of 12 authentic test sessions performed by 8 testers, a categorization of the knowledge types, summarized in Table 16, was described in detail. The detected failures were analyzed from two different viewpoints. First, the failure symptom classification, summarized in Table 17, was created to characterize the externally observable symptoms of the failures that would be recognizable to a tester and, thus, to the end-users of the software system. Second, the detection difficulty was analyzed using the FTFI-number, as presented in Table 18.

Table 16. Categories of knowledge used in detecting software failures (article V)

Knowledge category and perspective		Knowledge type
Domain knowledge	Users' perspective	Episodic knowledge of usage procedures and context
		Conceptual knowledge of the information content and presentation in the usage context
		Knowledge of problems in customer cases
	Application domain perspective	Conceptual knowledge of the subject matter
		Practical knowledge of the subject matter and tools
System knowledge	Interacting features and system perspective	Knowledge of the system's working mechanisms, logic, and interactions
		Knowledge of past failures
	Individual features and functional perspective	Knowledge of features and views of the system
		Knowledge of the detailed technical aspects
Generic knowledge	Generic correctness perspective	Knowledge of software user interfaces and presentation
	Usability perspective	Practical knowledge of the usability of software systems
	Direct failure perspective	Practical knowledge to recognize crashes and error messages

The analysis revealed three main types of experience that testers utilized to detect defects in the observed sessions: *domain knowledge, system knowledge, and generic SE knowledge*. Under each category, two or three perspectives were recognized. Domain knowledge was divided into the users' perspective and the application domain perspective, whereas system knowledge appeared as the interacting features and the system perspective and, on the other hand, individual features and functional perspective. The generic software engineering knowledge category was divided into three

perspectives, generic correctness, usability, and direct failure. The knowledge categories, perspectives, and specific knowledge types under each perspective are described in detail, with examples, in article V.

The results of this rigorous observation study show that the testers are capable of identifying failures without detailed test case descriptions of the expected outcomes. The testers identified failures based on their domain, system, and generic software engineering knowledge. The domain knowledge includes knowledge of users' needs and goals, and the system knowledge covers not only individual features but, even more importantly, the interactions of many features and the functioning of the system as a whole.

Experience-based knowledge is applied for testing in a distinctly different fashion compared to how the TCBT paradigm describes the software testing activity. The ways of applying knowledge in exploratory testing included evaluating the overall behavior of the system and comparing the features with other features and with knowledge of earlier versions. In many cases, knowledge was applied straightforwardly as a test oracle, but sometimes knowledge was applied as a more comprehensive strategy to guide testing. This comprehensive use of knowledge for detecting defects was identified in the data and called a test *wizard* to differentiate it from the use of knowledge as a test *oracle*. In these situations, knowledge was applied to design targeted attacks to address known risks or customer problems. The knowledge was also applied to generate the expected results as a part of the testing activity.

One of the most interesting findings of this study was that a significant share (20%) of identified failures in the study were revealed as side effects of the actual testing activity, which further emphasizes the diverse and creative opportunities of the exploratory testing approach.

Table 17. Failure symptom type classification (article V)

Commission failures	Presentation and layout
	Error message
	Extraneous functionality
	Inconsistent state
	Incorrect results
Omission failures	Data presentation and layout
	Missing function
	Lack of feedback
	Lack of capability

The failure symptom classification (Table 17) was created when analyzing the types of failures that were identified in the exploratory testing sessions. Suitable failure type classifications were not available in the literature,

which served as motivation to create a preliminary failure symptom classification based on this observation data. The resulting classification is based on a common dichotomy of omission and commission faults. In the failure symptom classification, this division has been applied at the highest-level classification. A finer granularity classification was created under the two main classes based on the observation data. This classification is an improvement of an earlier classification used in the experiment data analysis (see Table 11 in Section 4.2.2). A cross-analysis of symptom types and knowledge categories revealed that there were some failure types that seemed to be detected more often based on certain knowledge types. First, the presentation and layout as well as the error message failure symptoms were recognized mostly by generic knowledge. Second, the inconsistent status failure type seemed to require system knowledge. Third, incorrect results related strongly to domain knowledge. Fourth, the missing capability failure class was related to both domain and system knowledge.

This preliminary failure symptom classification adds to the body of empirical understanding of software failure types in real operational software systems. This classification can be used to guide testers and to create focused, failure-driven exploratory testing techniques. The classification increases the understanding of software failures from the viewpoint of the effects that the failures have on end-users. The classification is preliminary and needs to be further improved and extended with more failure data from different contexts.

Table 18. FTFI distribution among knowledge categories (article V)

FTFI number	Domain	System	Generic	Total
0 Dir. visible	5 %	12 %	4 %	8 %
1-way	75 %	34 %	44 %	45 %
2-way	15 %	37 %	19 %	26 %
3-way	5 %	5 %	7 %	7 %
Unclear	0 %	12 %	26 %	14 %

The FTFI distribution of the detected failures is presented in Table 18. This analysis shows that a clear majority of the identified failures fall into the directly visible or 1-way failure classes, meaning that the failures are straightforward to reveal. Only one-third of the failures fall into the 2-way or 3-way classes, meaning that there are two or three interacting variables, inputs, or actions involved in the occurrence of the failure. In comparing the knowledge categories, it seems that the failures related to domain knowledge are even more straightforward to reveal. The findings in this analysis suggest that the exploratory testing approach could be effective even when less experienced testers are used. On the other hand, the explor-

atory testing approach can be an effective way of involving the knowledge of domain experts in testing activities.

4.3.3 Testing contribution of different organizational groups

This research question is related to the significance of domain knowledge in software testing. System-level testing is traditionally seen as a separate action that is executed by independent testing specialists. However, there are several examples in our experience and hints in the literature that, in many cases, testing is actually a cross-cutting activity that involves knowledge and people from different organizational roles and functions. Based on the hypothesis of software testing as a cross-cutting activity, a study was designed to provide empirical evidence on how testing involves different groups of employees in varying organizational roles at software product development companies.

This research question was addressed by an exploratory case study in which the defect-reporting contributions of different organizational groups were studied in three software product development organizations. In this study, data was collected through interviews, defect database analysis, workshops, and informal communications with the company personnel.

Table 19. Distribution of defect reports between reporter groups (article VI)

	Case A		Case B		Case C		Total	
Sales & Consulting	145	8.7 %	117	8.5 %	136	7.3 %	398	8.1 %
Support	111	6.6 %	79	5.7 %	239	12.8 %	429	8.7 %
Manager	108	6.5 %	247	17.9 %	476	25.5 %	831	16.9 %
Internal Misc	348	20.8 %	89	6.4 %	620	33.2 %	1057	21.5 %
Specialized Tester	367	22.0 %	-	-	117	6.3 %	484	9.8 %
Developer	134	8.0 %	419	30.3 %	282	15.1 %	835	17.0 %
Customer (ext)	458	27.4 %	431	31.2 %	N/A	N/A	889	18.1 %
Total	1671		1382		1870		4923	

The results of the case study, reported in article VI, indicated that, at all three companies, employees from various groups participated in the software testing process, which is summarized in Table 19. The roles varied across the organization and, even if the responsibility for testing was assigned specifically to some group of employees, a wide variation in the people who reported defects was found. The existence of a separate testing group or hiring specialized testing consultants did not appear as a strong peak in the defect-reporting contribution. These results support the hypothesis that, in real software organizations, product testing is not a separate task of specialized testers. Instead, it seems to be a team activity in which a large number of people with varying organizational roles contribute and collaborate.

In terms of the differences in contribution between these organizational groups, the results lead to three findings. First, defects discovered by developers have the highest fix rate. Second, defects discovered by people whose main task is testing have the lowest fix rate. Third, people with a personal stake in the software product (e.g. due to a close connection to the customer) tend to set higher importance on the defects that they report. A summary of the number and importance of the defects and the fix rates across all three cases are presented in Figure 3.

In all cases, defects discovered by developers had the highest fix rate, even exceeding the fix rate of the defects detected by the customers, see Figure 3. One explanation for the high fix rate of developers' defects is that they fix their own defects, but, according to our further analysis, it seems that the self-reported defects only partially explain the developers' high fix ratios.

People with a personal stake in the product tend to place their defects at a higher level of importance than the company average, but it did not improve their defect fix ratio. This phenomenon was explained by two reasons. First, responsibility for a particular customer can explain higher defect importance, which means that customer satisfaction was directly related both to the defects and to the reporter's personal success in work. Second, the responsibility for a larger set of customers increases the personal stake and defect importance. People with close customer relationships tend to see the defects they report as higher priorities because the defects are more directly related to customer complaints or are based on their experience of what is important for the end users.

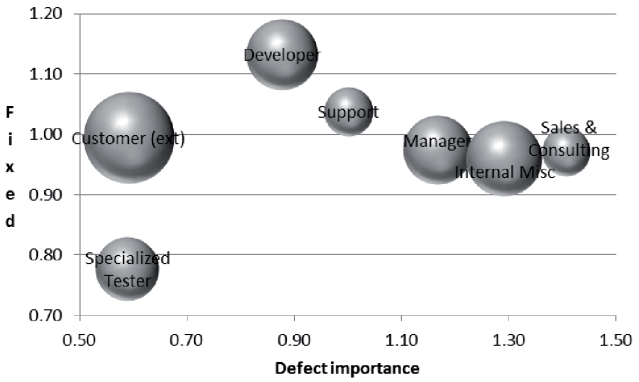


Figure 3. Bubble chart of defect importance (x-axis), fix rate (y-axis), and total number of defects found by each reporter group (indicated by the size of the bubble) in all cases. Data was normalized to company totals to enable cross-case merging (e.g., a value of 1.0 in both axes equals the companies' average). (article VI)

The major challenge of utilizing the people near the customer interface was to motivate and get these people involved in testing before the last-minute

demo presentations, which is far too late to have any defects that arise repaired by any usual processes. A solution to this challenge was identified in one case in which a practice called demo-by-customer was applied. In this practice, the (internal) customer was preparing a demonstration of the new features of an upcoming product release in an early phase. This effectively involved a person with domain knowledge in the testing activities.

Article VI also reports on what the case organizations value in software testing. The conclusion was that v*alidation from the viewpoint of end-users is more valuable than verification aiming for zero-defect software*. At all the companies, knowledge of the domain, the end-users, and the customer process was considered highly important in software testing.

4.3.4 Key findings

The third research goal was to gain empirical results indicating how exploratory testing is applied in industry and how knowledge is applied in ET. Based on the results of four studies (articles II, IV, V, and VI) summarized above, I would like to highlight the following key findings. First, several diverse practices of applying ET were identified in the case study. ET was not applied as a single testing technique; rather, it was an approach that was widely applied as part of more document-driven approaches and as an independent approach in varying forms. Important was the finding of ET as a common part of regression testing, which is a rather unorthodox approach.

Second, the recognition of exploratory and more systematic, manual testing practices provided new knowledge of how testers perform testing activities in practice. A finding that deviates from the conventional textbook testing approach was the application of comparison techniques as an oracle during test execution.

Third, a significant contribution was the study of the role of knowledge in defect detection in the context of ET (article V) and generally (article VI). New qualitative information was reported on the knowledge types applied in defect detection. The significance of domain knowledge in defect detection was showed qualitatively in the observation study, and the case study quantitative results show the important role of domain experts in defect detection. These results also highlight the applicability of the ET approach for involving domain expertise in software testing activities. In addition, the results of article VI showed the diverse organizational roles of people contributing to defect detection in software product organizations.

5. Discussion

In this section, the results are discussed and the answers to the research questions are presented. After that, the limitations of this thesis are presented, followed by the summary of the implications for researchers and practitioners.

5.1 Answers to the research questions

In this subsection, the results of this thesis are discussed and answers to the seven research questions are given. The discussion is structured according to the research goals stated in Section 3.1.

5.1.1 Definitions, motivation, and applicability ET

The first goal of this thesis was to *define exploratory testing and understand the applicability of ET based on literature review*. Two research questions were stated for this goal, and the answers to these questions are discussed next.

RQ 1: How is the ET approach defined in the literature?
The exploratory testing approach is described in the practitioner literature and consultant reports. As ET is an approach rather than a technique or methodology, the descriptions of it mainly describe the general properties of ET, not concrete practices or rules. Based on the available sources, five distinct characteristics of ET were identified in this thesis (see Section 4.1.1).

The definition is an important starting point for this research because it synthesizes the essence of the various descriptions of ET in books and experience reports. Because the ET approach is not widely known as a term, it is important to be able to describe it clearly. The definition is needed to understand what the main characteristics of this unstudied phenomenon are and to recognize the exploratory approach to testing in later empirical work.

RQ 2: How is ET motivated in the literature and in what contexts is ET claimed to be applicable?

The literature review indicated that ET is considered to be applicable in a wide range of contexts. In general, ET was seen as widely applicable as one complementary part of testing methods in most software testing contexts. In particular, certain contexts that are challenging for the TCBT approach were proposed as applicable for ET, such as when there is not enough time for detailed test case design, when rapid feedback and learning are required, when documented test cases cannot reveal any more defects, and situations in which the test cases are difficult to design in advance but are dynamically based on the results of previous tests. Another type of applicable context included situations that emphasize the strengths of the ET approach, as when the end-users' viewpoint is important in testing or when a particular risk or defect has to be investigated.

In practitioner reports, ET is motivated by the claimed benefits based on the authors' personal experiences. The proposed benefits are related to the effectiveness, in terms of the number and importance of found defects, and efficiency, in terms of the low amount of pre-design and documentation. In addition, simultaneous learning was proposed as a benefit. This is claimed to help testers come up with better and more powerful tests as testing proceeds. It is not possible to evaluate these claimed benefits based on the reports. The only way to gain a better understanding of those intuitively plausible claims is to conduct empirical studies.

The lack of statements concerning the shortcomings of ET was clearly a gap in the knowledge, especially considering the many intuitively obvious concerns of coverage and repeatability of tests as well as tracking and managing ET work. The applicability of ET was based on the authors' personal experiences and reasoned mostly by referring to the challenges and costs of the TCBT approach. Because of the lack of reliable research, it remained unclear what the consequences and drawbacks of the low level of documentation and relying on the experience of individual testers would be. Furthermore, based on the reviewed reports, it was not possible to understand what the actual factors were that made ET work in the reported cases. Was it the personal skills or knowledge of the people, the type of experience they had, their motivation, or some specific practices or processes?

The early steps of this thesis work included an analysis of the testing and QA practices of agile software development methodologies (article I). At that time, agile methodologies such as extreme programming (XP) (Beck 1999) proposed relying purely on automated testing approaches (Crispin & House 2003). In the results of the theoretical analysis (article I), several

shortcomings of testing practices in agile methodologies were identified. Many of these shortcomings were related to the fundamental challenges of test automation. The session-based ET approach was identified as an applicable improvement and, based on the results, we suggested including the exploratory testing approach to complement the automated testing and, thus, improve the testing practices in agile methods. Later, the applicability of ET in the agile context was recognized (Crispin & Gregory 2009, Whittaker 2009) and found to be beneficial (Tuomikoski & Tervonen 2009).

5.1.2 Benefits and shortcomings of ET

The second goal of this thesis was to *investigate the benefits and shortcomings of ET empirically*. Two research questions were stated for this goal, and the answers to these questions are discussed next.

RQ 3: What are the perceived benefits and shortcomings of ET in industry?

The perceived benefits of ET are twofold. First, the benefits are based on the abilities of the exploratory approach to work in realistic development contexts where documentation is weak, things are changing, and resources are limited. Second, practitioners clearly found the ET approach to be effective because it enables them to efficiently take advantage of the experience, knowledge, and creativity of a wide variety of people in the development organization. The perceived benefits were mainly similar to the ones that the literature review summarized, including the effectiveness, efficiency, and versatility of testing. A difference to the literature was that practitioners emphasized the difficulty and unacceptable amount of effort that the TCBT approach would require as the reason for using the ET approach. Other less-often covered benefits of ET in comparison to TCBT were the ability to form a better view of the overall quality of the tested software and learning the product for purposes other than testing, such as training and customer support.

In terms of the shortcomings of the ET approach, the results revealed that the interviewees found planning and measuring the coverage and the difficulty of tracking the testing work as the most serious shortcomings of the ET approach. In addition, for the case organizations, the dependency of individual testers and the lack of visibility to the actual testing work were clearly concerns when ET was utilized. The lack of repeatability of the tests and the reproducibility of found defects were not considered problems. These findings concerning the shortcomings of the ET approach are an im-

portant improvement to the published knowledge since the practitioner reports rarely cover the shortcomings of the approach.

An important finding was that, in all cases, the choice of ET was reasoned by the difficulties, complexities, and laboriousness of test case design. The ET approach, however, does not provide any specific techniques to tackle these challenges. Instead, it may just encourage people to ignore the complexities and make it easier to adapt to tight schedules because it does not make the amount and complexity of the tested functionality as explicit as detailed test designs would do. This aspect is reflected in the interviewees' concerns about controlling the coverage of ET and its sole reliance on individual testers' performance.

RQ 4: What is the defect detection effectiveness and efficiency of the ET approach in comparison to the TCBT approach?

In this thesis, the results concerning the defect detection effectiveness and efficiency of ET has been presented in two studies: the controlled student experiment (article III) and the case study (article II).

The controlled student experiment in article III focused on comparing the defect detection effectiveness of the ET and TCBT approaches. In the study, the subjects performed fixed-length testing sessions using both approaches. In the TCBT approach, the subjects performed a preparation phase, in addition to the testing session, during which they designed and documented the test cases for the actual testing sessions. The results showed that the ET approach detected slightly more defects but without a statistically significant difference between the approaches. However, the amount of preparatory test design work in the TCBT approach made the total effort in TCBT six times higher than in ET. This means that similar effectiveness was gained with much less effort in the ET approach. This comparison, however, is not completely fair because the many benefits of documented test cases related to managing and tracking testing work did not affect the results of this short experiment.

More insight into the efficiency of the ET approach is gained from the results of the case study of article II, in which the defect and effort data from two case organizations was available. This data showed that, in those organizations, ET produced 4.8 and 8.7 defects per testing hour when session-based ET was applied. This result is anecdotal in a sense that shows just a single metric from two organizations. It cannot be used to make conclusions as such, but, in comparison to some available efficiency data from experimental research, we can state that the efficiency numbers of session-based ET in this study seem to be high since, in two experiments that re-

ported efficiency data for TCBT (Wood et al. 1997, Andersson et al. 2003), both reported numbers below 3 defects per testing hour.

Based on these results, our hypothesis is that the ET approach is at least as effective in terms of defect detection as TCBT, but this is due to the lack of prescriptive test case design and much more efficient documentation. There are a few existing experiments that have compared ET or a similar approach to other testing methods, and the results of these studies also support this hypothesis (Houdek et al. 2002, do Nascimento & Machado 2007).

These results lead to the hypothesis that the benefits of test case documentation might be something other than prescribing the detailed test design, inputs, and expected results. Test case design techniques are constructed based on assumptions of typical defects, theories of coverage, and other concepts that aim at designing test cases that are effective at revealing defects and efficient in avoiding redundant testing. However, in ET, it seems that testers are capable of performing detailed test design and execution in parallel and without detailed prescription. In addition, the results of article III indicated that TCBT can lead to a much larger number of false reports, e.g., reporting duplicate or non-existent defects. If test case documentation does not actually help testers to reveal defects more effectively and can even damage the test results, test documentation should be produced to serve other purposes such as higher-level test planning, managing test projects, and tracking testers' work. In this case, the documentation could probably be lighter, be less detailed, and require less effort to produce and maintain in comparison to detailed test case documentation.

5.1.3 ET approaches in practice

The third goal of this thesis was to *provide empirically based results on how the ET approach is applied in practice*. Three research questions were stated for this goal, and the answers to these questions are discussed next.

RQ 5: How is the ET approach applied in industry?

The results of the case study in article II indicate that ET is not a single methodology or an approach with well-defined boundaries. Instead, many diverse ways of applying ET were identified. ET was applied both as an independent testing method and as a complementary part of TCBT methods. Several independent ET methods were identified, including session-based ET, functional testing of individual features, smoke testing, and subcontracted ET. A somewhat surprising finding was the use of the ET approach in regression testing, which was identified in two case organizations. Practi-

tioners found the personal experience of testers to be a valuable way of revealing defects related to interactions of fixes or changes and other features of the system. Exploratory testing was also applied regularly as a part of other development duties and as a complementary part of the TCBT approaches, e.g., to extend and deepen the testing beyond the documented test cases.

In another study based on field observations, in article IV, the practices of exploratory testers were identified. The results show that 22 testing practices were detected in the observations. These practices work as higher strategies guiding the testing work and, on the other hand, as detailed techniques for test execution, input selection, and comparison. These findings show that test execution in manual testing is not a mechanical activity but requires a lot of mental work. It also shows how exploratory testers use many strategies to manage the coverage and proceed systematically in their testing. When applying the ET approach, the testers use identifiable techniques in test design and input selection that are partly similar to the traditional test case design techniques. An important finding is the identification of the comparison techniques applied in ET as a test oracle.

Based on these results, it is clear that there are multiple ways of applying the ET approach as a part of the software development process. Later, Martin et al. (2007) and Rooksby et al. (2009) reported detailed ethnographic descriptions of real-world testing practices. These descriptions illustrate the real nature of testing activities and include a variety of examples in which exploratory ways of performing testing can be identified even though they do not use the actual term in their work.

In most of our cases, the development organizations learned the ET methods through practice by applying and modifying common testing techniques and approaches in their specific technical and organizational contexts. In article VI, the value of testing in the case organizations was studied, and the results showed that validation from the viewpoint of end-users is more valuable than verification and aiming for defect-free software. ET could be a more natural approach when the testing does not focus on the technical details but, instead, aims at ensuring validity and usefulness from the users' viewpoint. It seems that people in these organizations could not get useful advice from the literature or theory of software testing for their own contexts. It is important to describe and study these testing approaches and techniques that emerge from practice and start creating a body of ET knowledge that researchers and practitioners alike can utilize in future work.

RQ 6: How is knowledge applied to failure detection in exploratory testing in industry?

The second field observation study in this thesis focused on the role of testers' personal knowledge in failure detection. The results (article V) present a detailed qualitative analysis of 91 failure detection incidents observed by researchers in authentic testing situations. These results highlight the importance of testers' personal knowledge in failure detection when the ET approach is used. The results indicate how domain knowledge, system knowledge, and generic software engineering knowledge are applied in failure detection. The knowledge was applied mainly as a test oracle, i.e., for determining whether the result or behavior of the system under test was correct or not. The domain knowledge and system knowledge were also applied as a test wizard, meaning that the tester used his or her personal knowledge to guide testing and design tests targeted to attack a certain risk or known problem in the system. In addition, the analysis revealed that 20% of the detected defects in the observations were revealed as side effects. This means that defects were detected in different features from the actual target of the testing activity in question. This phenomenon is related to the nature of exploratory testing work where the tester is free to explore all of the functionality, encouraged to follow hunches, investigate suspicious behavior, and utilize personal knowledge of potentially risky features and interactions. The large share of side-effect failures supports the assumption that exploratory testing is more versatile in comparison to TCBT. The testers are capable of covering a larger set of features and investigating the software with a wider scope than what is instructed in the actual testing task.

The types of detected defects in the field study were also analyzed. The failures were classified based on the visible symptoms and difficulty of detection. Certain failure types seemed to be typically detected based on a specific knowledge type. Based on the analysis of the interrelationships of knowledge types and failure types, the following four hypotheses were generated. 1) In ET, testers are able to utilize their personal knowledge of the application domain, the users' needs, and the tested system for defect detection. 2) In ET, testers frequently recognize relevant failures in a wider set of features than the actual target features of the testing activity. 3) A large number of the failures in software applications and systems can be detected without detailed test design or descriptions. 4) The majority of failures related to domain knowledge are straightforward to recognize, and failures related to system knowledge or generic software development knowledge are more complicated to recognize in terms of the number of interactions.

The testers' ability to detect failures without pre-described expected results again puts the role of test documentation into a new light. In traditional test case documentation, the detailed description of the expected results is essential (IEEE 2008). Specifying the expected results is, in practice, challenging and requires a lot of effort, which means that if, in certain contexts, testers are capable of detecting failures without such descriptions, it would be a considerable improvement in efficiency, as proposed in the earlier results of this thesis.

The role of personal knowledge partly explains the differences in the ET and TCBT approaches. The results presented in this thesis give rich descriptions of how defect detection happens in ET work and why testers are capable of detecting large numbers of failures without detailed descriptions of the expected results or test inputs and steps. In the exploratory approach, testers seem to focus not only on the functioning of individual features but take a wider view of the functioning of the system as a whole and the interactions of different features. Testers in ET consider the usage context, including the tasks and goals of the end-users when testing, and do not restrict their evaluation only to the technical features of the system.

More importantly, these results show that ET can be applied to exploit domain expertise in software testing directly, without the need to codify the knowledge first in the form of documented test cases. This is an important result because the domain knowledge is found to be an important factor in testing (Beer & Ramler 2008, Iivonen et al. 2010, Merkel & Kanij 2010), but involving domain experts from outside testing organization into TCBT activities can be too difficult. ET methods can provide a more straightforward means to engage people with varying knowledge in testing.

RQ 7: How do people in different organizational roles contribute to defect detection?

During the empirical research, in multiple software development organizations, we experienced that testing, particularly through the exploratory approach, was not purely an activity of professional testers. It was clear that many organizational groups were performing testing activities in one form or another. This was, however, a phenomenon that was not directly discussed in the literature. We found some studies that described testing in real organizations that seemed to support our initial observations. The last study of this thesis (article VI) analyses the contributions of different organizational groups in a detailed case study. The results very clearly show the significant contribution of diverse organizational groups to defect detection. The main findings indicate that people close to the customers and with a personal stake in the product have a high contribution in terms of both the

number and importance of the reported defects. The high importance values of their defects do not, however, improve their fix rate. Instead, developer's defects had the highest fix rates, while specialized testers had both low fix rates and low importance values.

In the context of organizations' developing highly technical software products, defect detection contribution is distributed to diverse organizational groups, and the contribution of specialized testers does not stand out in terms of the number, importance, or fix rates of the defects. While the high importance of experience and domain knowledge to testing has been identified in the literature (Beer & Ramler 2008, Iivonen et al. 2010, Kettunen et al. 2010, Merkel & Kanij 2010), the results of this thesis show quantitatively how much people with different knowledge in different organizational roles, in practice, contribute to defect detection. In addition, considering the hypothesis of the applicability of the ET approach to involve domain experts in testing, the results of this case study emphasize the relevance of and need for such an approach. Because of the high number and diversity of people contributing to defect detection, testing approaches that are applicable for involving these groups in testing are needed.

5.2 Validity threats

This section contains a summary of the main limitations and threats to the validity of this research. Descriptions of the limitations related to the details of each individual study are presented in the articles. This thesis employed three different research methods: case studies, a controlled experiment, and field observations. Each of the methods had their own threats to validity, and they are described in this section using common terminology employed in experimental software engineering (Wohlin et al. 2000) and case study research (Yin 1994, Runeson & Höst 2009). In the next subsections, internal and conclusion validity, reliability, construct validity, and external validity of this research are covered.

5.2.1 Internal and conclusion validity

The internal and conclusion validity are relevant for experimental research, so in this thesis they concern the controlled experiment study (article III). Threats to conclusion validity are issues that affect the ability to draw conclusions based on experimental data (Wohlin et al. 2000). In the experiment the number of subjects was sufficient for statistical analysis, but the reliability of measures could be a threat to conclusion validity. The measures were based on the subjects' own reports of their findings, so there

might be differences between what the subjects reported and what they actually found during the experiment. In addition, the individual differences in how subjects applied the exploratory testing approach in the experiment caused some level of uncontrollable variation in the experiment results.

Threats to internal validity are factors other than the treatment, unknown to the researcher, that can affect the independent variable with respect to causality (Yin 1994, Wohlin et al. 2000, Runeson & Höst 2009). The training and education provided during the course probably affected how the student subjects applied both of the studied testing approaches. In particular, it was not possible to completely prevent subjects from applying exploratory activities during test case based testing. Even though the subjects were randomly assigned to treatment groups and each subject applied both approaches in the experiment, it is possible that an unknown factor exists that has a stronger effect on results than the testing approach that was used as a treatment.

5.2.2 Reliability

In the qualitative case and field studies in this thesis, the goals were exploratory and descriptive, and theory generating rather than hypotheses verifying. In this context, the validity criteria are different, and the concepts related to conclusion validity and internal validity that focus on the relationship between treatment and outcome and causality are not applicable. Instead, the reliability of the research can be assessed via the repeatability of the data collection and analysis procedures (Yin 1994, Runeson & Höst 2009), i.e., how strong the researcher's influence is and how much the results are dependent on specific researchers.

In the field observation studies of this thesis, an important threat is the possible subjective interpretations of the observing researcher. This was a serious threat in the first field study in which the observations were not recorded. In the second field study, the video recording made it possible to more objectively perform the analysis based on the primary data. In the case studies, the reliability was ensured by using multiple data sources and more than one researcher in the data collection and analysis. However, the researchers definitely influenced the results. First, in case studies during semi-structured interviews, the interviewer unavoidably set the questions and direction of the discussion based on his or her personal experience and viewpoints. Second, in data analysis of the field studies, the approach was exploratory and the method was applied grounded theory, which means that the codes and findings emerge from the data without an existing theory

or coding scheme. This type of analysis is highly dependent on the researcher performing the analysis.

5.2.3 Construct validity

Threats to construct validity are issues related to the operationalization of the theoretical concepts in an empirical study, i.e., how well the measured data represents the theory and concepts under study (Yin 1994, Wohlin et al. 2000, Runeson & Höst 2009). The most important threats to construct validity were related to the rather vague concept of ET. In the first case study (article II), the main threat to construct validity was the interviewees' interpretation of the ET approach. Interviewees were hesitant to describe their true (exploratory) testing practices because they felt that they were not real or acceptable ways of performing testing. ET was a term not familiar to all interviewees, which forced the interviewers to describe it in the interview situation. In the student experiment (article III), it is possible that the actual techniques that subjects performed varied between subjects, which also is a threat to construct validity. Construct validity is also affected by the interaction of testing and the treatment. Both in the experiment and two field studies the effect of being evaluated or observed could have affected the behavior of the subject. In the student experiment, it is possible that the knowledge that they would be graded biased the behavior of the subjects. In the field observation studies, observing most probably affected subjects' behavior. However, it seems that the observer's presence affected more the attention and focus of the tester than the actual practices. It seemed that the testers were more focused, had fewer interruptions, and identified more defects when they were observed by the researcher.

5.2.4 External validity

Threats to external validity are issues concerning the generalization of results from a single experiment or case study to a wider, more general population or other organizations in industry (Yin 1994, Wohlin et al. 2000, Runeson & Höst 2009). The most important threats to external validity of this thesis are, first, the effect of the context both in the student experiment and in the industrial studies and, second, the sampling strategies used to select the industrial organizations and individual subjects for the studies.

Using students as subjects in the experiment affects the external validity of the results. We cannot know how well students represent professional testers. It is possible that one of the compared approaches suffers more from the inexperience of student subjects. Another threat is how well the

short testing sessions of the experiment compare to the typical way of working in industry.

All the industrial studies of this thesis are limited to a similar context, which means that generalizing the results to very different software systems or development contexts is not possible. The main limiting context variables in the case and field studies of this thesis are the following. The studied companies were small to medium-sized enterprises; the development organizations had fewer than 100 employees; all organizations were in the software product development business; the developed software products were targeted for professional users, not consumer markets; products were highly interactive systems with rich graphical user interfaces; and the products or product lines were relatively mature, with years or decades of development history. In addition, all the organizations were in Finland, which means that the results are not necessarily applicable to organizations located in countries with a very different culture.

The sampling of case organizations was based mainly on accessibility through research cooperation. Some selection among the available organizations was made in the individual studies and, in the first field study (article IV), one organization was included from outside the research project participants. Most of the case companies had organized software testing as part of other development units, not as independent testing organizations, which probably limits the generalizability of the results in organizations with clearly separate testing organizations.

The selection of individuals for interviews and observations was guided by the research design, but the practical availability of people at the companies affected the selection.

5.3 Implications for research

The studies in this thesis open a new research area in the field of software testing. This work introduces the exploratory testing approach to the research community and motivates its relevance by empirical studies in industry. In addition, a first step in the empirical investigation of the effectiveness and efficiency of ET is taken, and the experimental design and results are documented for further studies.

For the research community, the field studies provide valuable qualitative data on low-level exploratory testing practices and testers' behavior, which give explanations and insight into the proposed benefits and working mechanisms of the ET approach.

A major implication is the detailed analysis of the role of knowledge in exploratory testing work and the stated hypotheses that are grounded in empirical data for future research to test. Unlike most of the related research, which relies on interview data, this study investigates and describes the role of knowledge in detail and is grounded in primary observation data and authentic failure detection incidents.

This work makes a methodological contribution by studying software engineering practices using video-based field observations and applied grounded theory methods for data analysis. This combination of methods has rarely been used in software engineering research. Direct observations clearly have benefits as a data collection method in studying software development activities, but there are challenges in collecting and analyzing such data. This study describes our method of performing data collection and analysis and, thus, provides valuable advice for other researchers in the software engineering field.

5.4 Implications for practice

This thesis identifies the ET approach, which seems to be practically relevant but is largely ignored in the literature. The results outline the benefits and applicability of the ET approach and also discuss the shortcomings. This information is helpful for practitioners who consider applying the ET approach and need an understanding and results concerning these issues to support their decision-making.

The ET practices and approaches described in the case and field studies can be used as concrete guidance and as examples of how ET could be carried out and applied as a part of other testing activities.

In the practitioner literature, there are some claims regarding the effectiveness and efficiency of the ET approach. This thesis provides empirical results that support at least the comparable effectiveness and better efficiency of ET in comparison to TCBT. Along with a few other studies that report similar findings, this thesis strengthens the available evidence regarding the effectiveness and efficiency of the ET approach. Such evidence is important for practitioners when considering suitable testing approaches in a certain context and improvements to existing testing methods.

The analysis of the role of personal knowledge in defect detection showed that defects can be and are detected based on different types of knowledge. For practitioners, this means that one should consider what type of application and system knowledge will be important for their products and how this knowledge could be exploited in testing. The findings also suggest that

the ET approach could be an effective way of engaging people with varying types of personal knowledge in testing activities.

I propose that practitioners recognize the ET approach as a valid testing approach and consider its benefits among other testing methods. This research indicates that ET is a beneficial approach, at least in the functional testing of interactive software products with rich user interfaces from the end users' point of view.

6. Conclusions and future work

In this thesis, the exploratory software testing approach has been studied using empirical research methods. This section states the contributions and conclusions of this research. Finally, directions for future work are outlined in the second subsection.

6.1 Contributions of the research

This research makes five main contributions. First, this thesis initiated the research on ET approaches in the software engineering research community. While the ET approach has been known to practitioners and found in textbooks, this thesis is the first research work that specifically focuses on ET.

Second, the experimental study on the defect detection effectiveness provided the first scientific results on the effectiveness of the ET approach. This study raised the hypothesis that ET can be as effective as TCBT but require less effort, and it provided initial empirical evidence for it.

Third, the empirical studies in industry provided new knowledge of how and why ET is applied in practice and how defect detection in general is an organizationally cross-cutting activity instead of the responsibility of specialized testing organizations. This work draws a picture of this testing approach that is distinctively different from the more traditional paradigm of an independent testing organization performing TCBT. This work analyses the testing approach on both the individual and organizational levels.

Fourth, the field studies on ET practices and the role of knowledge in defect detection reported a detailed analysis of the testing activities in authentic manual testing situations. Manual testing is a rarely studied topic and, in particular, the test execution and defect detection activities have been addressed in no previous scientific research. The results make a highly valuable contribution to understanding and classifying the knowledge that testers need to detect failures during testing. In addition to the knowledge classification, the types of detected failures were analyzed, which gives a richer view of the ET activity and the role of knowledge in it.

Fifth, this research stated hypotheses regarding the effectiveness of the ET approach, the role of knowledge in failure detection, and the types of failures that are detected in ET. These hypotheses are grounded in empirical data and serve as starting points for future studies.

As a final conclusion, I state that exploratory software testing is a relevant and applicable approach among practitioners. In scientific literature, however, the ET approach has not been recognized or studied. This research has taken the first steps to begin evaluating the ET approach using empirical research methods. The results suggest that ET can be as effective as test case-based approaches and even more efficient. The role of testers' personal knowledge is important in failure detection, and ET was found to be an applicable approach to engage domain experts in testing.

6.2 Future work

The effectiveness and efficiency of exploratory testing approach requires much more research and evidence to enable the drawing of reliable conclusions. In particular, important studies would be empirical evaluations in an industrial context.

Research on the methods of managing, planning, and tracking ET is needed. To understand the real benefits and limitations of ET, it should be studied as a complete testing methodology that can be applied in properly managed software engineering projects and processes. It is important to study how the practitioners can benefit from the strengths of ET without sacrificing other important aspects of testing, such as planning, tracking and coverage.

The ET approach is found to be applicable and effective in industry, but there are several challenges and shortcomings that have not been solved and need research. The applicability and context dependencies of ET are not well-understood and are one important research area.

There is a fundamental need to study detailed ET practices and techniques. An important question is how much the experience-based aspects of ET can be captured in codified form and taught or trained to novice testers or, e.g., domain experts who possess the relevant personal knowledge but are not testing experts.

Finally, this research posed several hypotheses that should be tested and updated, if needed, in future studies.

References for the summary

Aberdour, M., 2007. Achieving Quality in Open Source Software. *IEEE Software*, 24(1): 58-64.

Abran, A., Moore, J.W., Bourque, P., Dupuis, R. & Tripp, L.L., 2004. *Guide to the Software Engineering Body of Knowledge 2004 Version*. IEEE Computer Society, Los Alamitos, CA, USA.

Adelson, B. & Soloway, E., 1985. The Role of Domain Experience in Software Design. *IEEE Transactions on Software Engineering*, 11(11): 1351-1360.

Andersson, C. & Runeson, P., 2002. Verification and validation in industry - a qualitative survey on the state of practice, in *Proceedings of International Symposium on Empirical Software Engineering*. pp. 37-47.

Andersson, C., Thelin, T., Runeson, P. & Dzamashvili, N., 2003. An Experimental Evaluation of Inspection and Testing for Detection of Design Faults, in *Proceedings of International Symposium on Empirical Software Engineering*. pp. 174-184.

Bach, J., 1999. General Functionality and Stability Test Procedure for Certified for Microsoft Windows Logo. Available at: http://www.satisfice.com/tools/procedure.pdf

Bach, J., 2000. Session-Based Test Management. *Software Testing and Quality Engineering*, 2(6).

Bach, J., 2004. Exploratory Testing, in van Veenendaal, E. (Ed.), *The Testing Practitioner*. UTN Publishers, Den Bosch, pp. 253-265.

Baresi, L. & Young, M., 2001. *Test Oracles* (Technical Report, No. CISTR-01-02). University of Oregon, Eugene, Oregon, USA.

Basili, V.R. & Selby, R.W., 1987. Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering*, 13(12): 1278-1296.

Beck, K., 1999. Embracing Change With Extreme Programming. *Computer*, 32(10): 70-77.

Beer, A. & Ramler, R., 2008. The Role of Experience in Software Testing Practice, in *Proceedings of Euromicro Conference on Software Engineering and Advanced Applications*. pp. 258-265.

Beizer, B., 1990. *Software Testing Techniques*. Van Nostrand Reinhold, New York.

Berner, S., Weber, R. & Keller, R.K., 2005. Observations and Lessons Learned from Automated Testing, in *Proceedings of International Conference on Software Engineering*. pp. 571-579.

Bhatti, K. & Ghazi, A.N., 2010. *Effectiveness of Exploratory Testing, An empirical scrutiny of the challenges and factors affecting the defect detection efficiency* (Master's Thesis). Blekinge Institute of Technology, Ronneby, Sweden.

Bolton, M., 2005. Testing Without a Map. *Better Software*.

Burnett, M., Cook, C. & Rothermel, G., 2004. End-user software engineering. *Communications of the ACM*, 47: 53–58.

Cockburn, A., 2002. *Agile Software Development*. Addison-Wesley, Boston.

Copeland, L., 2004. *A Practitioner's Guide to Software Test Design*. Artech House Publishers, Boston.

Craig, R.D. & Jaskiel, S.P., 2002. *Systematic Software Testing*. Artech House Publishers, Boston.

Creswell, J.W., Clark, V.L.P., Gutmann, M.L. & Hanson, W.E., 2003. Advanced Mixed Methods research Designs, in Tashakkori, A., Teddlie, C. (Eds.), *Handbook of mixed methods in social & behavioral research*. SAGE.

Crispin, L. & Gregory, J., 2009. *Agile testing: a practical guide for testers and agile teams*. Addison-Wesley, Boston.

Crispin, L. & House, T., 2003. *Testing Extreme Programming*. Addison-Wesley, Boston.

Cross, N., 2004. Expertise in design: an overview. *Design Studies*, 25(5): 427-441.

Dallas, A., 2010. Caution: V&V May Be Hazardous to Software Quality. *Medical Device & Diagnostic Industry*, 32(5).

Engelke, C. & Olivier, D., 2002. Putting Human Factors Engineering Into Practice. *Medical Device & Diagnostic Industry*, 24(7).

Engström, E. & Runeson, P., 2010. A qualitative survey of regression testing practices, in *Proceedings of International Conference on Product-Focused Software Process Improvement*. pp. 3-16.

Fewster, M. & Graham, D., 1999. *Software Test Automation*. Addison-Wesley, Harlow, England.

Forward, A. & Lethbridge, T.C., 2002. The relevance of software documentation, tools and technologies: a survey, in *Proceedings of the ACM Symposium on Document Engineering*. pp. 26-33.

Fowler, M. & Highsmith, J., 2001. The Agile Manifesto. *Software Development*, 9(8): 28-32.

Følstad, A., 2007. Work-Domain Experts as Evaluators: Usability Inspection of Domain-Specific Work-Support Systems. *International Journal of Human-Computer Interaction*, 22(3): 217.

Galletta, D.F., Abraham, D., El Louadi, M., Lekse, W., Pollalis, Y.A. & Sampler, J.L., 1993. An empirical study of spreadsheet error-finding performance. *Accounting, Management and Information Technologies*, 3(2): 79-95.

Galletta, D.F., Hartzel, K.S., Johnson, S., Joseph, J. & Rustagi, S., 1996. An Experimental Study of Spreadsheet Presentation and Error Detection, in *Proceedings of Hawaii International Conference on System Sciences*. pp. 336-345.

Hellmann, T.D., 2010. *Enhancing Exploratory Testing with Rule-Based Verification* (Master's Thesis). University of Calgary, Calgary, Alberta, Canada.

Houdek, F., Schwinn, T. & Ernst, D., 2002. Defect Detection for Executable Specifications — An Experiment. *International Journal of Software Engineering and Knowledge Engineering*, 12(6): 637-655.

Howden, W.E., 1978. Theoretical and Empirical Studies of Program Testing. *IEEE Transactions on Software Engineering*, 4(4): 293-298.

Hughes, J. & Parkes, S., 2003. Trends in the use of verbal protocol analysis in software engineering research. *Behaviour & Information Technology*, 22(2): 127.

Hulkkonen, E., 2010. *Mobiiliohjelmistojen tutkiva testaus* (Master's Thesis). Tampere University of Technology, Tampere.

Höfer, A., 2008. Video analysis of pair programming, in *Proceedings of the International Workshop on Scrutinizing Agile Practices or Shoot-out at the Agile Corral*. pp. 37-41.

IEEE, 2008. *IEEE Standard for Software and System Test Documentation* (Standard, No. IEEE Std 829-2008). IEEE Computer Society, New York.

Iivonen, J., Mäntylä, M.V. & Itkonen, J., 2010. Characteristics of high performing testers: a case study, in *Proceedings of International Symposium on Empirical Software Engineering and Measurement*. p. 60:1.

Janzen, D. & Saiedian, H., 2005. Test-driven development concepts, taxonomy, and future direction. *Computer*, 38(9): 43-50.

Juristo, N. & Moreno, A.M., 2001. *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, Boston.

Juristo, N., Moreno, A.M. & Vegas, S., 2004. Reviewing 25 years of Testing Technique Experiments. *Empirical Software Engineering*, 9(1-2): 7-44.

Juristo, N., Moreno, A.M., Vegas, S. & Shull, F., 2009. A Look at 25 Years of Data. *Software, IEEE*, 26(1): 15-17.

Kaner, C., Bach, J. & Pettichord, B., 2002. *Lessons Learned in Software Testing*. John Wiley & Sons, Inc., New York.

Kaner, C., Falk, J. & Nguyen, H.Q., 1999. *Testing Computer Software*. John Wiley & Sons, Inc., New York.

Kasurinen, J., Taipale, O. & Smolander, K., 2010. Test case selection and prioritization: risk-based or design-based?, in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*. pp. 10:1–10:10.

Kettunen, V., Kasurinen, J., Taipale, O. & Smolander, K., 2010. A study on agility and testing processes in software organizations, in *Proceedings of the International Symposium on Software Testing and Analysis*. pp. 231-240.

Kharlamov, M., Polovinkin, A., Kondrateva, E. & Lobachev, A., 2008. Beyond Brute Force: Testing Financial Software. *IT Professional*, 10(3): 14-18.

Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., Emam, K.E. & Rosenberg, J., 2002. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8): 721-734.

Lethbridge, T.C., Sim, S.E. & Singer, J., 2005. Studying Software Engineers: Data Collection Techniques for Software Field Studies. *Empirical Software Engineering*, 10(3): 311-341.

Lyndsay, J. & van Eeden, N., 2003. Adventures in Session-Based Testing. Available at: http://www.workroomproductions.com/papers/AiSBTv1.2.pdf

Martin, D., Rooksby, J., Rouncefield, M. & Sommerville, I., 2007. "Good" Organisational Reasons for "Bad" Software Testing: An Ethnographic Study of Testing in a Small Software Company,

in *Proceedings of International Conference on Software Engineering*. pp. 602-611.

Memon, A., Banerjee, I. & Nagarajan, A., 2003. What test oracle should I use for effective GUI testing?, in *Proceedings of International Conference on Automated Software Engineering*. pp. 164-173.

Merkel, R. & Kanij, T., 2010. *Does the Individual Matter in Software Testing?* (Technical Report, No. 2010-001). Swinburne University of Technology, Centre for Software Analysis and Testing.

Miles, M.B. & Huberman, M.A., 1994. *Qualitative Data Analysis: An Expanded Sourcebook*. SAGE Publications, Thousand Oaks.

Myers, G.J., 1979. *The Art of Software Testing*. John Wiley & Sons, New York.

do Nascimento, L.H.O. & Machado, P.D.L., 2007. An experimental evaluation of approaches to feature testing in the mobile phone applications domain, in *Proceedings of the Workshop on Domain Specific Approaches to Software Test Automation*. pp. 27–33.

Naseer, A. & Zulfiqar, M., 2010. *Investigating Exploratory Testing in Industrial Practice* (Master's Thesis). Blekinge Institute of Technology, Rönneby, Sweden.

Ng, S.P., Murnane, T., Reed, K., Grant, D. & Chen, T.Y., 2004. A preliminary survey on software testing practices in Australia, in *Proceedings of the Australian Software Engineering Conference*. pp. 116-125.

van Niekerk, J.C. & Roode, J.D., 2009. Glaserian and Straussian grounded theory: similar or completely different?, in *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*. pp. 96–103.

Page, A., Johnston, K. & Rollison, B., 2008. *How We Test Software at Microsoft*. Microsoft Press.

Patton, M.Q., 2002. *Qualitative Research and Evaluation Methods*, 3rd ed. Sage, Thousand Oaks.

Persson, C. & Yilmaztürk, N., 2004. Establishment of Automated Regression Testing at ABB: Industrial Experience Report on "Avoiding the Pitfalls", in *Proceedings of the 19th International Conference on Automated Software Engineering*. pp. 112-121.

Phalgune, A., Kissinger, C., Burnett, M., Cook, C., Beckwith, L. & Ruthruff, J.R., 2005. Garbage in, Garbage out? An Empirical Look at Oracle Mistakes by End-User Programmers, in *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. pp. 45-52.

Pichler, J. & Ramler, R., 2008. How to Test the Intangible Properties of Graphical User Interfaces?, in *Proceedings of 1st International Conference on Software Testing, Verification, and Validation*. pp. 494-497.

Poon, P.-L., Tse, T.H., Tang, S.-F. & Kuo, F.-C., 2011. Contributions of tester experience and a checklist guideline to the identifica-

tion of categories and choices for software testing. *Software Quality Journal*, 19(1): 141-163.

Rautiainen, K., 2004. *Cycles of Control: A Temporal Pacing Framework for Software Product Development Management* (Licentiate Thesis). Helsinki University of Technology, Espoo, Finland.

Rooksby, J., Rouncefield, M. & Sommerville, I., 2009. Testing in the Wild: The Social and Organisational Dimensions of Real World Practice. *Computer Supported Cooperative Work*, 18(5-6): 559-580.

Runeson, P., 2006. A survey of unit testing practices. *IEEE Software*, 23(4): 22-29.

Runeson, P. & Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2): 131-164.

Ruthruff, J.R., Burnett, M. & Rothermel, G., 2005. An empirical study of fault localization for end-user programmers, in *Proceedings of the International Conference on Software Engineering*. pp. 352–361.

Salinger, S., Plonka, L. & Prechelt, L., 2008. A Coding Scheme Development Methodology Using Grounded Theory for Qualitative Analysis of Pair Programming. *Human Technology*, 4(1): 9-25.

Sandberg, J., 2000. Understanding human competence at work: An interpretative approach. *Academy of Management Journal*, 43(1): 9-25.

Saukkoriipi, S., 2010. *Defining and utilizing team exploratory testig sessions* (Master's Thesis). University of Oulu, Oulu, Finland.

Seaman, C.B., 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4): 557-572.

Seaman, C.B. & Basili, V.R., 1998. Communication and organization: an empirical study of discussion in inspection meetings. *IEEE Transactions on Software Engineering*, 24(7): 559-572.

Shah, S.M.A. & Alvi, U.S., 2010. *A Mix Testing Process Integrating Two Manual Testing Approaches: Exploratory Testing and Test Case Based Testing* (Master's Thesis). Blekinge Institute of Technology, Rönneby, Sweden.

Shahamiri, S.R., Kadir, W.M.N.W., Ibrahim, S. & Hashim, S.Z.M., 2011. An Automated Framework For Software Test Oracle. *Information and Software Technology*, 53(7): 774-788.

Shoaib, L., Nadeem, A. & Akbar, A., 2009. An empirical evaluation of the influence of human personality on exploratory software testing, in *Proceedings of IEEE International Multitopic Conference*. pp. 1-6.

Shull, F., Singer, J. & Sjøberg, D.I.K. (Eds.), 2008. *Guide to Advanced Empirical Software Engineering*. Springer London, London.

Sonnentag, S., 1998. Expertise in professional software design: A process study. *Journal of Applied Psychology*, 83(5): 703-715.

Stebbins, R.A., 2001. *Exploratory Research in the Social Sciences*. SAGE Publications, Thousand Oaks.

Strauss, A.L. & Corbin, J.M., 1998. *Basics of qualitative research: techniques and procedures for developing grounded theory*. SAGE.

Tinkham, A. & Kaner, C., 2003a. Exploring Exploratory Testing, in *Proceedings of the Software Testing Analysis & Review Conference*. p. 9.

Tinkham, A. & Kaner, C., 2003b. Learning Styles and Exploratory Testing, in *Proceedings of the Pacific Northwest Software Quality Conference*.

Tuomikoski, J. & Tervonen, I., 2009. Absorbing software testing into the scrum method, in *Proceedings of 10th International Conference on Product-Focused Software Process Improvement*.

Turley, R.T. & Bieman, J.M., 1995. Competencies of exceptional and nonexceptional software engineers. *Journal of Systems and Software*, 28(1): 19-38.

Våga, J. & Amland, S., 2002. Managing High-Speed Web Testing, in Meyerhoff, D., Laibarra, B., van der Pouw Kraan, R., Wallet, A. (Eds.), *Software Quality and Software Testing in Internet Times*. Springer-Verlag, Berlin, pp. 23-30.

Weyuker, E.J., 1982. On Testing Non-Testable Programs. *The Computer Journal*, 25(4): 465 -470.

Whittaker, J.A., 2000. What is Software Testing? And Why is it so Hard? *IEEE Software*, 17(1): 70-79.

Whittaker, J.A., 2003. *How to Break Software A Practical Guide to Testing*. Addison Wesley, Boston.

Whittaker, J.A., 2009. *Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design*. Addison-Wesley Professional.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B. & Wesslén, A., 2000. *Experimentation in software engineering: an Introduction*. Kluwer Academic Publishers, Boston, MA, USA.

Wood, B. & James, D., 2003. Applying Session-Based Testing to Medical Software. *Medical Device & Diagnostic Industry*, 25(5): 90.

Wood, M., Roper, M., Brooks, A. & Miller, J., 1997. Comparing and combining software defect detection techniques: a replicated empirical study. *ACM SIGSOFT Software Engineering Notes*, 22(6): 262-277.

Wu, H., Guo, Y. & Seaman, C.B., 2009. Analyzing Video Data: A Study of Programming Behavior under Two Software Engineering Paradigms, in *Proceedings of International Symposium on Empirical Software Engineering and Measurement*. pp. 456-459.

Yin, R.K., 1994. *Case Study Research: Design and Methods*. Sage Publications, Inc.

Part II: Articles

I **Toward an Understanding of Quality Assurance in Agile Software Development**
Juha Itkonen, Kristian Rautiainen, and Casper Lassenius
Published in *International Journal of Agile Manufacturing*, 2005, vol 8, no. 2: 39–49.

II **Exploratory Testing: A Multiple Case Study**
Juha Itkonen and Kristian Rautiainen
Published in *Proceedings of International Symposium on Empirical Software Engineering*, 2005, pp. 84–93.

III **Defect Detection Efficiency: Test Case Based vs. Exploratory Testing**
Juha Itkonen, Mika V. Mäntylä, and Casper Lassenius
Published in *Proceedings of International Symposium on Empirical Software Engineering and Measurement*, 2007, pp. 61–70.

IV **How Do Testers Do It? An Exploratory Study on Manual Testing Practices**
Juha Itkonen, Mika V. Mäntylä, and Casper Lassenius
Published in *Proceedings of International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 494–497.

V **The Role of Knowledge in Failure Detection During Exploratory Software Testing**
Juha Itkonen, Mika V. Mäntylä, and Casper Lassenius
Submitted to *IEEE Transactions on Software Engineering*, May 2011, 17 pages.

VI **Who Tested My Software? Testing as an Organizationally Cross-Cutting Activity**
Mika V. Mäntylä, Juha Itkonen, and Joonas Iivonen
Published in *Software Quality Journal*, published online 21st August 2011, 28 pages.

Exploratory software testing is an experience-based approach to revealing defects. It relies on the tester's knowledge and skills, and is based on creative exploration instead of comprehensive test documentation. Thus, it is a fundamentally different approach than the traditional test-case-based testing paradigm. Exploratory testing is commonly used in software organizations as practitioners consider it an effective and efficient approach for detecting defects. Exploratory testing is also considered as an effective way of involving application domain expertise in testing. Despite this practical relevance, very little scientific research on exploratory testing exists. This dissertation opens up a new research path and provides the first empirical results on the applicability and effects of exploratory testing based on qualitative and quantitative empirical studies.

Aalto-DD 107/2011

ISBN 978-952-60-4339-5 (pdf)
ISBN 978-952-60-4338-8
ISSN-L 1799-4934
ISSN 1799-4942 (pdf)
ISSN 1799-4934

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

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS