Mälardalen University Press Licentiate Theses
No. 136

SOFTWARE TESTING IN AGILE DEVELOPMENT

TECHNOLOGICAL AND ORGANISATIONAL CHALLENGES

Adnan Čaušević

2011

MÄLARDALEN UNIVERSITY
SWEDEN

School of Innovation, Design and Engineering

ISBN 978-91-7485-015-4
ISSN 1651-9256
Printed by Mälardalen University, Västerås, Sweden

Abstract

The contemporary industrial trend towards agile software development brings forth new concerns, challenges as well as opportunities. One of the main issues concerns the quality of the final product, for which testing is the well-known assurance mechanism. However, how to perform testing using existing expertise in an agile environment presents a challenging issue for the software industry. This can potentially create confusion and contra productivity which can lead to a situation where testing teams and their practices are considered obstacles for the full implementation of agile processes within an organisation.

This thesis identifies and addresses test-related organisational and technological challenges in an agile environment. In this context, we propose a new role for traditional testers which enables them to integrate with the agile team as well as fully exploit their knowledge in the new context. We have conducted an elaborate industrial survey on the preferences and practices with respect to the contemporary aspects of software testing, and identified test-driven development as an important technological area for improvement. A subsequently performed systematic review on empirical evidence related to test-driven development revealed a list of factors which may limit its widespread industrial acceptance and usage. Knowledge of testing was identified as one of those factors and we further attempted to confirm its significance through a controlled experiment performed with master students.

Our future works aim to confirm these research findings in wider as well as industrial settings, and investigate other limiting factors in detail, with the aim of providing guidelines for achieving better utilisation of testers and testing practices.

Acknowledgements

This thesis could not have been done without the great support of my supervisor Sasikumar Punnekkat and my co-supervisors Daniel Sundmark and Ivica Crnković. Thank you guys for your leadership, patience and knowledge you shared so unselfishly. Even though this thesis was my destination, your supervision made me realise how the journey itself mattered the most.

As a Ph.D. student I was relying on my supervisors support in publishing research results, but co-authoring with researchers out of my comfort zone greatly improved my collaboration and interaction skills. Indeed this is something I am very thankful for to Abdulkadir Sajeev, Rikard Land, Frank Lüders, and Iva Krasteva.

Travelling to conferences and research project meetings is another link in the chain of experience a graduate student should have. Thank you Stig Larsson, Sigrid Eldh, and Radu Dobrin for being an often travel companion in this phase of my study. Mingling is so much easier with you guys around.

When not travelling, I had to share my office space with really great roommates: Srinivasan Jayakanth (JK), Stefan Björnander, Kathrin Dannmann, Etienne Borde, Aleksandar Dimov, Andreas Johnsen, Vijayalakshmi Saravanan (Viji), Hüseyin Aysan, Abhilash Thekkilakattil, and Jiale Zhou. Thank you guys for being silent, but also cheerful and always ready for a small talk.

It's not very easy to focus on the research when there are administrative issues hanging above your head. Luckily, I had administrative people around me to always rely on. Thank you Harriet Ekwall, Gunnar Widforss, Monica Wasell, Susanne Fronnå, Carola Ryttersson, and Malin Rosqvist.

Directly or indirectly, many senior researchers at MDH have provided help to my Ph.D. studies. Thank you Hans Hansson, Kristina Lundqvist, Paul Pettersson, Cristina Seceleanu, Thomas Nolte, Dag Nyström, Damir Isović, Jan Carlson, and Tiberiu Seceleanu mostly for isolating me from the world of funding but also for having the time for me and my questions.

As a person I am very dependant on the communication and interaction with other human beings, and MDH could not be a better choice to get many interesting discussions during the coffee breaks, travels or other social events. Thank you Ana Petričić, Ana Živković, Aneta Vulgarakis, Antonio Cicchetti, Barbara Gallina, Batu Akan, Branka Pavetić, Farhang Nemati, Federico Ciccozzi, Giacomo Spampinato, Hongyu Pei-Breivold, Jagadish Suryadevara, Josip Maraš, Juraj Feljan, Leo Hatvani, Luka Lednički, Mehrdad Saadatmand, Mikael Åsberg, Moris Behnam, Nikola Petrović, Rafia Inam, Saad Mubeen, Séverine Sentilles, Stefan Bygde, Svetlana Girs, Thomas Leveque, and Yue Lu for sharing a few moments of your life with me.

I would like to express my gratitude to my parents Zuhdija and Šefika Čaušević as well as to my sister Azra Čaušević for their unconditional support and love through all of my life. I appreciate your smile and understand your tears.

I would like to thank to my wife Aida Čaušević for supporting me and believing that I can achieve much more. If I have to start this journey again I could not imagine anyone else beside me except you. I love you!

And last, but most certainly not the least, I would like to thank to my daughter Alina Čaušević for making me a complete person. Your smile, your bite, your hug, your cry... everything of yours helps me move forward. I love you, too.

Adnan Čaušević
Västerås, June 21, 2011

List of Publications

Papers Included in the Licentiate Thesis[1]

Paper A *An Industrial Survey on Contemporary Aspects of Software Testing*, Adnan Čaušević, Daniel Sundmark and Sasikumar Punnekkat, In proceedings of the International Conference on Software Testing (ICST), Paris, France, April 2010

Paper B *Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review*, Adnan Čaušević, Daniel Sundmark and Sasikumar Punnekkat, In proceedings of the International Conference on Software Testing (ICST), Berlin, Germany, March 2011

Paper C *Impact of Test Design Technique Knowledge on Test Driven Development: A Controlled Experiment*, Adnan Čaušević, Daniel Sundmark and Sasikumar Punnekkat, In submission

Paper D *Redefining the role of testers in organisational transition to agile methodologies*, Adnan Čaušević, A.S.M. Sajeev and Sasikumar Punnekkat, In proceedings of International Conference on Software, Services & Semantic Technologies (S3T), Sofia, Bulgaria, October, 2009

[1]The included articles are reformatted to comply with the licentiate thesis specifications

Other relevant publications

Conferences, Workshops and Poster Sessions

- *Reuse with Software Components - A Survey of Industrial State of Practice*, Rikard Land, Daniel Sundmark, Frank Lüders, Iva Krasteva and Adnan Čaušević, International Conference on Software Reuse, Springer, Falls Church, VA, USA, September, 2009

- *A Survey on Industrial Software Engineering*, Adnan Čaušević, Iva Krasteva , Rikard Land, A.S.M. Sajeev and Daniel Sundmark, Poster session at International Conference on Agile Processes and eXtreme Programming in Software Engineering (XP2009), p 240241, Springer, Sardinia, Italy, Editor(s):P. Abrahamsson, M. Marchesi, and F. Maurer, May, 2009

Technical Reports

- *An Industrial Survey on Software Process Practices, Preferences and Methods*, Adnan Čaušević, Iva Krasteva , Rikard Land, A.S.M. Sajeev and Daniel Sundmark, MRTC report ISSN 1404-3041 ISRN MDH-MRTC-233/2009-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, March, 2009

Contents

I

Thesis

Chapter 1

Introduction

Traditional software development life cycle has become inadequate to preserve quality of software products when organisations attempt to shorten their time-to-market. In many cases the quality control is often reduced or postponed due to the reduced deadlines or overrun of the development phase [1] [2]. Organisations are in need of a new process that will value quality in each stage of their product development without interfering with the product delivery schedule. They are increasingly turning their interest to agile methodologies [3].

Agile is, indeed, a software development philosophy that will battle with short delivery schedules by creating a product with fewer features instead of lowering quality standards of the same product. The problem is that many proven cases of agile development in large scale environment are specific to each organisational setting and their best practices cannot be easily implemented within another organisation. Of course, at the same time, we can only guess the number of unsuccessful agile development attempts in organisations, without publicly available reports on their failures (in literature known as publication bias [4]). However, during our involvement in FLEXI, an EU-ITEA2 funded Project [5], we became aware from our industrial partners, of many of issues related to the transition from the traditional lifecycle to the Agile approach. One of the reason for such issues, could be in fact that organisations are trying to reuse techniques and tools from traditional development process that may not be applicable within particular agile practices, and blamely Agile development processes may not be fully justifiable.

The research presented in this thesis, originated from such a premise and investigates if traditional approaches to software testing with existing practices in place could be utilised to full extent within agile development.

1.1 Background

In this thesis we will be using several concepts from three different areas, viz., Agile development, Software testing and Test-Driven Development. We now present some key concepts from these areas, before providing the details on the contributions of this thesis.

1.1.1 Agile Development

Agile development is considered a relatively young software engineering discipline that emerged from industrial needs for a software development process where the main focus should be on the customer and their business needs. The idea is to have a constant communication channel with the customer by iteratively providing working software product with currently most needed business values built in. Historically, the idea behind an agile approach is actually not new. It was reported [6] that NASA Project Mercury (first US human spaceflight program in 1960s) used time-boxed iterations with tests written before each increment - an activity very similar to what is known today as a test-driven development (TDD).

Agile is not a software development process by definition, but rather a philosophy based on a set of principles. These principles are listed in the so called "Agile Manifesto" [7]. Since understanding of agile is relying on those twelve principles, we are listing them here:

1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3. *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
4. *Business people and developers must work together daily throughout the project.*

5. *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7. *Working software is the primary measure of progress.*
8. *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9. *Continuous attention to technical excellence and good design enhances agility.*
10. *Simplicity - the art of maximizing the amount of work not done - is essential.*
11. *The best architectures, requirements, and designs emerge from self-organizing teams.*
12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

Agile Manifesto [7]

By following these principles, organisations are committing to have a continuous feedback with customer and provide value to their business needs.

Several software development processes use some of those principles, like: eXtreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Feature Driven Development (FDD), etc. usually referring to them as *agile software development methods*. Aside from following agile principles, each of those methods contains different *agile practices*. Pair programming (PP), test-driven development (TDD) and continuous integration (CI) are just a few to mention.

An overview of one Scrum iteration (sprint), as an example of agile development process, is shown in Figure 1.1. Prioritised product backlog is used to select user stories for the upcoming sprint. By dividing them into concrete tasks, they become part of the current sprint backlog. During the period of 2-4 weeks only items in the current sprint are completed on a daily basis. After each sprint a potentially shippable product increment should exist.

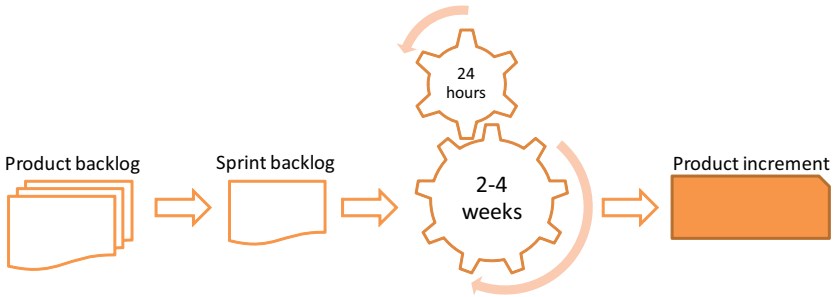

Figure 1.1: One sprint overview in Scrum process

1.1.2 Software Testing

Software testing is a major activity in software development and has two main goals:

- **to confirm** a software solution is behaving as per its requirements, and
- **to find faults** in a software which are leading to its misbehaviour.

It is important to note how testing cannot be used as a proof of fault free software. A famous quote from Edsger Dijkstra [8] is describing this as: "Testing can only show the presence of errors, not their absence". One of the reasons why we cannot claim there are no faults in software is in fact that exhaustive testing of any, especially complex systems, is just not possible due to the high number of variables influencing its final outcome.

Commonly, there are three levels of testing of software systems [9]:

- **System level** - has the purpose of testing overall system functioning from a user perspective.
- **Integration level** - has the purpose of testing interconnections between various components/modules during their integration phase.
- **Unit level** - has the purpose of testing functional and non-functional properties of a single unit/module/component of the system.

Software testing is a widely researched domain of its own with a multitude of techniques and tools proposed for industrial practice. A comprehensive discussion on this vast research domain is beyond the scope of this thesis and hence not attempted.

1.1.3 Test-driven development

Test-driven development (TDD), sometimes referred as test-first programming, is a practice within the extreme programming development method proposed by Kent Beck [10]. TDD requires the developers to construct automated unit tests in the form of assertions to define code requirements before writing the code itself. In this process, developers evolve the systems through cycles of testing, development and refactoring. This process is shown in Figure 1.2.

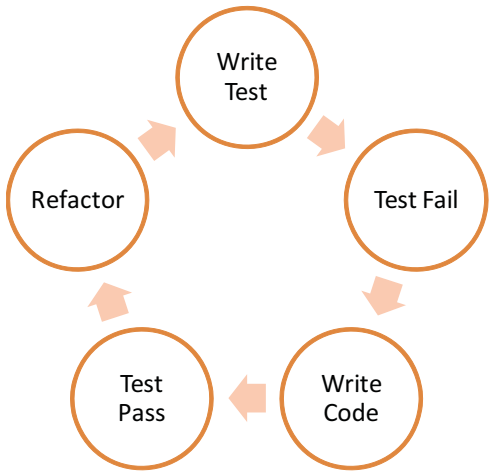

Figure 1.2: Test-driven development practice overview

In their experiment, Flohr and Schneider [11] prescribed TDD activities to students as a list of next activities:

1. *Write one single test-case*
2. *Run this test-case. If it fails continue with step 3. If the test-case succeeds, continue with step 1.*
3. *Implement the minimal code to make the test-case run*
4. *Run the test-case again. If it fails again, continue with step 3. If the test-case succeeds, continue with step 5.*
5. *Refactor the implementation to achieve the simplest design possible.*

> 6. *Run the test-case again, to verify that the refactored implementation still succeeds the test-case. If it fails, continue with step 5. If the test-case succeeds, continue with step 1, if there are still requirements left in the specification.*
>
> Flohr and Schneider [11]

1.2 Motivation and Problem Description

Today's business needs are demanding from software organisations to accept a constant pace of change as it reflects the current market and economic demands. According to the agile philosophy delivering an evolving software product without having a predefined set of requirements that will be changed at a later stage is something companies should not fight against, but rather embrace. Agile software development is one representative of the current industrial solutions to this challenge.

But this comes with a price. Adopting agile development for many organisations creates not only a phase shift in thinking on how to develop software but it also introduces significant amount of changes to their daily activities [12]. These changes consist of facilitating continuous product integration, ability to prioritise tasks, committing to its delivery all the way through daily stand-up meetings and burn-down charts.

In particular, changes affecting testing teams and testers may create additional confusion with respect to understanding who is responsible for th product quality and how to allocate time for this activity. In agile development, quality is everyone's responsibility and having in mind that traditional testing can consume even more than 50% of the total development time [9], testers do have a concern of ensuring how this time will be allocated in agile development.

1.3 Outline of thesis

This thesis consists of two main parts. The first part is organised as follows: Chapter 2 presents a summary of the research conducted with description of the research process and its major contributions. Chapter 3 provides related work with respect to both technological and organisational perspectives of our research. Thesis conclusion and guidelines for future work are outlined in Chapter 4. The second part of the thesis consists of Chapters 5 through 8 which represent research publications included in this thesis.

Chapter 2

Research Summary

Overall goal of our research efforts is:

to identify deficiencies in current testing practices in agile development environments and provide validated methods of better utilization of testers and testing techniques.

In order to help organisations successfully utilise agile practices, we set out to investigate how well software testing fits with the state of the practice of agile philosophy or the agile manifesto. The goal of this research could be viewed from two dimensions:

- Technological, defined with the top-level research question:
  RQ-1: *What are the technological challenges of traditional software testing in agile?*
- Organisational, defined with the top-level research question:
  RQ-2: *What are the organisational challenges of traditional software testing in agile?*

From the technological perspective, the goal is to identify test related practices, methods, techniques, improvements or practice adoptions which will provide most benefit to an organisation. It is also required to identify limiting factors for usage of such practices in an industrial environment.

From an organisational or process point of view, the goal is to define a new role for testers during an organisational transition towards agile methodology. It is our belief that this role will enhance the stature of testers as well as enable the company to effectively deploy the testers in the new environment.

2.1 Research Methodology

The research is based on empirical methodologies including analysis of qualitative and quantitative data. Literature and industrial surveys were performed in order to perceive the state of the art and state of the practice. Experiences from industry on this topic were collected and summarised with the research in a reusable form on a higher level of abstraction intended to be provided as guidelines for transition organisations.

2.2 Research Process

In Figure 2.1 an overview of the conducted research process is presented.

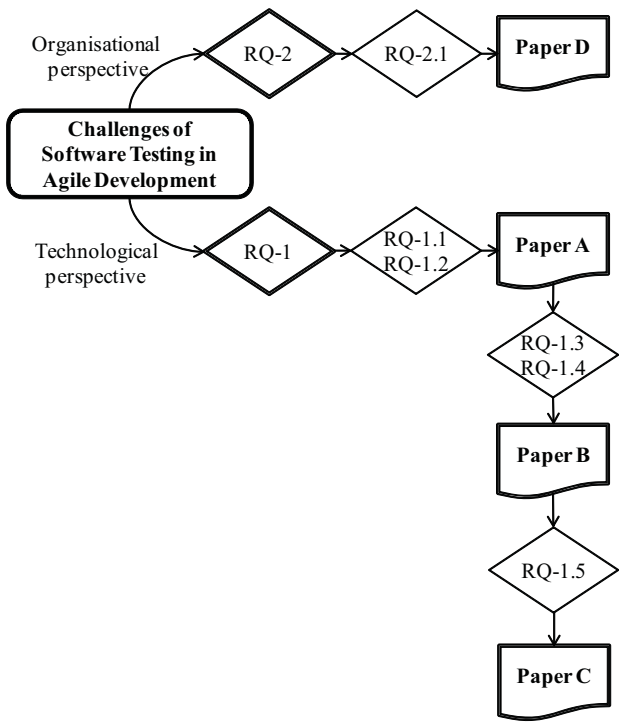

Figure 2.1: Research process overview

2.2.1 Technological perspective

As a starting point in detailed investigation of the top-level research question within the technological perspective (RQ-1), we decided to start the process by forming next research question:

RQ-1.1: *What are the current industrial preferences and practices related to the contemporary trends on software testing?*

To address this question, we decided to join our effort with several other researchers in order to define and execute a questionnaire through an online web based survey [13]. With this survey we specifically targeted industrial opinion on the usage of the current and preferred industrial practices and methods on software testing. During the formulation, execution and analysis of this empirical study, the subsequent research question evolved as:

RQ-1.2: *Can we identify the factor in which the preference and practice show maximum difference?*

After analysis phase was performed on the collected data, out of 22 examined test related practices, test-driven development (TDD) gained the highest score of "dissatisfaction". This means that among the respondents, the accumulated absolute difference between the preferred and the actual level of usage of TDD was highest. Further analysis revealed that the preferred level of usage of TDD was significantly higher than the actual level at which it has been practised. This result was interpreted as "Respondents would like to use TDD to a significantly higher extent than they actually do presently". This was an interesting finding for which we could not provide any clear and obvious reasons why this situation exist in industry. In order to get the broader view of the problems related to usage of TDD, the next research question was formulated as:

RQ-1.3: *How can we get a deeper insight on the factor with maximum difference?*

Realising that TDD as a practice should be investigated further we had to make a decision on how to proceed with the research process. One alternative was to further investigate industrial opinions by performing directed interviews with selected organisations. Another could be to organise a new questionnaire

survey with specific and directed questions relating to the usage of TDD. The problem with those solutions was that they are all providing an industrial perspective to the usage of TDD which we to some extent already gained from our first survey. We thought that academic opinions on the usage of TDD should also be considered in our research since after looking at some initial search results we noticed a growing number of empirical publications directly investigating benefits of TDD. For those reasons we decided to perform a systematic literature review on empirical studies of TDD.

When completed, the systematic literature review brought forward a list of 48 empirical studies on TDD, conducted in academic, industrial or mixed settings. Study participants were students as well as professionals. This result lead to forming a new research question:

RQ-1.4: *Can we identify and list limiting factors of TDD from the results of the literature study?*

Empirical studies, identified in the systematic literature review, were performed with different experiment designs (number of participants, complexity of problems, duration of study, etc.) making it difficult to directly compare the findings and easily create a common conclusion. We decided to identify and list all negative, neutral or positive effects of or on TDD and group them in common effect areas. Especially, we noted effects of TDD with explicit claims on requirements for a successful usage of TDD. In order for effect area to be considered as a limiting factor, next criteria had to be fulfilled:

- The effect area had to contain at least two studies with observations of negative effects of or on TDD
- The effect area had to contain more studies with observations of negative effects of or on TDD than studies with observations of positive effects of or on TDD
- Negative effects in the effect area had to be observed in at least one study performed in an industrial setting

Applying those criteria on selected research publications identified and listed seven potential limiting factors of industrial adoption of TDD: increased development time, insufficient TDD experience/knowledge, lack of upfront design, domain and tool specific issues, lack of developer skill in writing test cases, insufficient adherence to TDD protocol, and legacy code.

Out of these seven factors, we decided to explore one factor in detail to confirm its impact and see what kind of guidelines could be provided. “Lack of testing knowledge” came as the first obvious choice due to our own research leanings as well as due to the potential for independent exploration and perceived impact. The next research question was formed as:

RQ-1.5: *Can we confirm significance of testing knowledge as a limiting factor for TDD adoption?*

During the autumn of 2010 a controlled experiment with master students was performed as part of the course on Software Verification and Validation provided by Mälardalen University. The objective of the experiment was to investigate if developers who were educated on general testing knowledge will be able to utilise TDD more effectively. As a result of the experiment we noticed that students had difficulties creating negative test cases.

2.2.2 Organisational perspective

In order to perform detailed investigation of the top-level research question within the organisational perspective (RQ-2), we setup the next specific research question:

RQ-2.1: *What to do with traditional testing department when an organisation transits to agile development process, where tester’s roles seems to be ambiguous and diminished?*

In this investigation we considered several options for traditional software testers during their organisation’s transition towards agile software development. Among various alternatives we proposed a new role of: “Project Mentor” for testers. With this role we wanted to emphasise testers ability to communicate with development team on technical aspects of software development while at the same time being able to recognise the value for the customer by understanding the overall functional behaviour of the system.

2.3 Contribution

Since the thesis is written as a collection of papers, its contributions are summarised with contributions from each individual research paper. Relation between research paper contribution and research questions is presented in Table 2.1.

	Paper A	Paper B	Paper C	Paper D
RQ-1	✓	✓	✓	
RQ-1.1	✓			
RQ-1.2	✓			
RQ-1.3		✓		
RQ-1.4		✓		
RQ-1.5			✓	
RQ-2				✓
RQ-2.1				✓

Table 2.1: Relation between research questions and publications

2.3.1 Paper A

An Industrial Survey on Contemporary Aspects of Software Testing, Adnan Čaušević, Daniel Sundmark and Sasikumar Punnekkat, In proceedings of the International Conference on Software Testing (ICST), Paris, France, April 2010

Summary Using data from an industrial survey [13] a state of the practice paper was written. The survey in addition to confirming some popular beliefs also lists several noteworthy findings from the perspectives of respondent categories such as safety-criticality, agility, distribution of development, and application domain. These findings clearly depict negative discrepancies between the current practices and the perceptions of the respondents. This paper covers RQ-1.1 and provide contribution to RQ-1.2 by identifying test-driven development (TDD) as a factor with maximum difference between current and preferred practice.

My contribution I was the main author of this paper contributing with data analysis (performed using custom made software, developed by me for this

purpose). Co-authors supervised the process and helped in formulating findings and descriptive statistics.

2.3.2 Paper B

Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review, Adnan Čaušević, Daniel Sundmark and Sasikumar Punnekkat, In proceedings of the International Conference on Software Testing (ICST), Berlin, Germany, March 2011

Summary As a direct result of investigation from Paper A, a systematic literature review on TDD was performed. After initial keyword search on seven major research databases, results yielded 9462 publications. In several steps we removed publications that are not of an interest having 48 publications as the final number of our systematic review. With this activity RQ-1.3 was addressed. The process of extracting effects areas on or of TDD from selected research publications and identifying limiting factors contributed to RQ-1.4. Seven limiting factors were identified viz., increased development time, insufficient TDD experience/knowledge, lack of upfront design, domain and tool specific issues, lack of developer skill in writing test cases, insufficient adherence to TDD protocol, and legacy code.

My contribution I was the main author of this paper contributing in obtaining collection of papers from the search databases, filtering and removal as well as analysis of findings presented in selected collection of papers. Co-authors helped to filter the papers and also performed reading of selected list of publications to validate the findings.

2.3.3 Paper C

Impact of Test Design Technique Knowledge on Test Driven Development: A Controlled Experiment, Adnan Čaušević, Daniel Sundmark and Sasikumar Punnekkat, (In submission)

Summary Among the seven limiting factors identified from the systematic study in Paper B, knowledge of testing was selected to be further investigated as part of a controlled experiment with master students in order to address research question RQ-1.5. The experiment was designed around course on Software Verification and Validation at Mälardalen University. Participants were

divided into two groups solving two problems on two different occasions, before and after the course. The analysis was performed on the collected source code and test scripts created by students, as well as questionnaire survey responses. Results are showing positive improvements of test code coverage but no statistically significant difference exist between pre- and post- course groups. Qualitative analysis of data revealed lack of negative test cases resulting in students inability to detect bugs related to unspecified behaviours.

My contribution I was the main author of the paper, contributing in setting up the pre-requirements for the experiment (lab instructions, problems user stories, SVN, etc.), collecting data points and performing the analysis. Co-authors helped in study design, analysis of the data and in writing section on statistical analysis.

2.3.4 Paper D

Redefining the role of testers in organisational transition to agile methodologies, Adnan Čaušević, A.S.M. Sajeev and Sasikumar Punnekkat, In proceedings of International Conference on Software, Services & Semantic Technologies (S3T), Sofia, Bulgaria, October, 2009

Summary This paper provides a state of the art analysis of tester role in Agile organisation and propose a new role called “Project Mentor”. A major task of project mentors is to manage the expectations of the customers and other stake holders. This requires domain knowledge and the ability to speak in the language of the customers, which often programmers lack. Similarly, for managers, recognising the limitations of programmers is also a difficult task. Managers without a technical background often fail to understand difficulties which are faced by programmers on a daily basis. Testers as project mentors, we believe, will be in a position to better appreciate these difficulties and translate them to other stake holders with the help of their domain knowledge. A mentor’s role of helping others to implement quality in their daily activities could contribute significantly to the success of the project. This paper directly address research question RQ-2.1.

My contribution Idea for this paper originated from a discussion with visiting professor Abdulkadir Sajeev. I was the main author of this paper but the writing process was an iterative contribution of all authors.

Chapter 3

Related Work

Since our research is based on challenges from two fairly different perspectives, technological and organisational, we are presenting here related work from both of them independently.

3.1 Technological perspective

Agile does not have a formal definition behind its processes which makes it very hard for academic researchers to measure the quality impacts it can produce and in specific to reason about its claimed success. What researchers can do is to perform a series of empirical studies in academic or industrial settings for the purpose of evaluating quality improvements introduced with agile methodologies. Another aspect of investigation about agile development are the growing number of claimed success stories from industry that are presented to the community. By contributing with their experience and lessons learnt from projects with varying size and duration, industry is making a significant impact on the current body of knowledge that should not be neglected.

The central research paper on agile methodologies is ”Empirical Studies of Agile Software Development: A Systematic Review” [14]. This systematic literature review provides information regarding up to date findings w.r.t. empirical evidence of agile software development. It also provided additional insights for our own systematic literature review of empirical studies on TDD. Another additional resource on general understanding of agile methods is a chapter of Williams [15] within Advances in Computers book series where she describes different agile principles, practices and methodologies.

3.1.1 Empirical Studies on TDD

Several publications with empirical finding were also used in our research. In this section we are grouping them by the aim of the study itself.

Benefits of TDD

Müller & Hagner [16] performed an experiment with students divided into two groups, test-first and traditional, with focuses on the programming efficiency, the reliability of the resultant code and program understanding. Flohr & Schneider [11] had an experiment with students divided into two groups (test-first and classical-test) for the purpose of investigating impact of test-first development process. Gupta & Jalote [17] performed an experiment with students divided in two groups (TDD and waterfall) evaluating the impact of TDD on designing, coding, and testing. Data is obtained by questionnaire and forms. Kollanus & Isomöttönen [18] performed experiment with students on understanding TDD and perception on difficulties of TDD. Data was collected by questionnaire.

Quality of produced code

George & Williams [19] had professional developers from three companies in TDD and waterfall-like control groups to investigate code quality improvements. Another controlled experiment of Janzen & Saiedian [20] examined the effects of TDD on internal software design quality. The experiment was conducted with undergraduate students in a software engineering course. Janzen et al. [21] had empirical studies in three industry short courses investigating effects of test-driven development (TDD) on internal software quality. Vu et al. [22] performed an experiment with students divided in two experimental groups (test-first and test-last) in a year-long software engineering course evaluating productivity, internal and external quality of the product, and the perception of the methodology.

Productivity improvements

Geras et al. [23] executed experiment with professional developers divided in two groups working on two problems using test-first and test-last processes to investigate productivity and software quality. Huang & Holcombe [24] had a controlled experiment with students that investigated the distinctions between the effectiveness of test-first and test-last approaches.

Quality of tests

Erdogmus et al. [25] performed an experiment with undergraduate students divided into two groups (test-first and test-last) investigating test per unit effort, quality and productivity. Madeyski [26] had an experiment with students divided in test-first and test-last groups examining branch coverage and mutation score indicator of unit tests.

Impact of experience

Müller & Höfer [27] investigated conformance to TDD of professionals and novice TDD developers. Höfer & Philipp [28] performed an experiment with professionals and students investigating if expert programmers conform to TDD to a higher extent than novice developers.

3.1.2 Test-related research

One of the key papers on software testing is: "A Survey on Testing Technique Empirical Studies: How Limited is our Knowledge" [29]. This paper provides a valuable analysis of maturity level of the knowledge on testing techniques. Several research activities with the focus on agile and testing are also identified in literature. Schooenderwoert et al. [30] are discussing different agile test techniques for embedded systems while Paige et al. [31] are creating discussion around extreme programming development for high integrity systems. Eunha et al. [32] are describing a test automation framework for agile development and testing with more focus on the developer side of testing.

3.2 Organisational perspective

A seminal document for agile development is the "Agile Manifesto" [7] explaining the main agile principles and goals behind its philosophy. This document represents a main point in our investigation on how to adopt the process while still conforming to the agile principles. By looking into some industrial reports it is possible to see how IBM is transitioning their team to agile [33], how Microsoft [34] is overcoming communication problems with testers or how the Israeli Air Force [35] is adding value to their team by introducing an outside professional tester. Some organisations are even willing to share their lessons learnt from mistakes in adopting agile [36].

3.2.1 Transitioning to Agile

We are relating our work with two approaches from the organisational perspective on how to address the role of testers issue while transitioning to agile development. Sumrell [37] reports on the experience in transitioning from Waterfall to Scrum. One of the major issues was to decide how to transform the QA team and their testing strategies to the new environment. The approach taken for the QA team is to continue to have the primary responsibility of testing, but share it with developers and project managers. Instead of testers waiting until the parts are ready for test, the new approach would be a quicker build cycle so that the QA team can do its work rather than having to wait. Retraining is needed for QA personnel to be able to instrument code for testing rather than rely on previous practices of automated testing strategies. However, unit testing becomes largely the responsibility of the developers. We can identify several characteristics of this approach. One, the role of tester is somewhat diminished because some of the testing is now done by the developer. The tester requires retraining on the technical side. The tester needs to work more closely with developers and project managers thus requiring a higher level of group working skills. We hypothesise that in such an environment, a tester needs to be given adequate training for this transition, otherwise, it is likely that he or she will fail in the new environment where they are not in control of quality, and becomes just another member of a team.

Gregory and Crispin [38] discuss in detail the role of testers in agile development. Their recommendation is to make testers a part of the development team. The role of testers is to help clarify customer requirements, turn them into tests, and help developers understand the customer requirements better. Testers need to speak the domain language of the customer and the technical language of the developers. The characteristics of this approach include an increased role for testers as the link between customers and developers in addition to their role of testing. It is a shift in their work environment as they move from the Quality Assurance Division to be part of development pairs or groups. They probably will need retraining on interpersonal skills to work closely with customers and developers more than they are used to in the past.

Chapter 4

Conclusions and Future Work

This thesis represents a set of activities conducted as part of a research process in order to identify and address potential challenges of software testing in agile development. By performing various empirical studies (questionnaire survey, literature review and controlled experiment) we brought upfront test-driven development as a noteworthy testing research direction, investigating why this practice is not utilised to a higher extent within industrial settings.

During our investigation of the current body of knowledge, we identified 18 effect areas out of which 7 are considered as limiting factors on the industrial adoption of TDD, namely, increased development time, insufficient TDD experience/knowledge, lack of upfront design, domain- and tool-specific issues, lack of developer skill in writing test cases, insufficient adherence to TDD protocol, and legacy code.

We set up a controlled experiment with master students to investigate if developers knowledge of testing can affect adoption of TDD. Two groups of students were using TDD to solve two juxtaposing problems before and after the course on Software Verification and Validation. It is noticeable that code coverage increased in both groups after the course, but we could not identify any statistically significant difference between the groups. Further analysis of students achievements revealed lack of test cases with the focus on negative testing.

From an organisational perspective of agile adoption, we investigated possible options for transition of traditional testers into an agile environment. We

propose to define a new role for testers called “Project Mentor” which will emphasise their understanding of the complete system from a user perspective, but also utilise their technical knowledge in communication with developers.

In summary, the main contributions of this thesis are:

- The identification of TDD as a practice with most dissatisfaction in industry
- Listing seven potentially limiting factors for industrial adoption of TDD
- Pointing out student’s inability to write negative test cases during controlled experiment
- Proposing the need for augmenting the TDD with the new process steps or specific testing knowledge
- Proposing the “Project Mentor” role for traditional testers in an agile environment

Concerning future work, the process of identifying limiting factors for industrial adoption of TDD was conducted using peer-reviewed scientific publications that have been addressing validity threats of their empirical study. In order to confirm significance of identified limiting factors our future work will focus on obtaining insights from industrial reports which were not covered in our previous study due to the validity requirements. This will be done in combination with industrial interviews to cover the full scope of obstacles for full utilisation of test-driven development approach.

As indicated by our study, TDD also needs to be supplemented with new process steps or test design techniques, which could potentially further enhance the robustness and the reliability of the system. In this context, we will investigate how TDD can be augmented for achieving improved code quality while keeping its fundamental principles.

In a long term research perspective, we also intent to perform an industrial case study investigating how experienced developers could benefit from testing knowledge and what kind of specific testing knowledge they need in order to increase the quality of the code artefacts they produce.

Apart from conforming the existing contributions of our research, our future work will focus on approaching as close as possible to the goal set up at the very beginning of our research:

to identify deficiencies in current testing practices in agile development environments and provide validated methods of better utilization of testers and testing techniques.

Bibliography

[1] Annual Testing Survey. Technical Report Suite 350, Quality Assurance Institute, 7575 Dr. Phillips Blvd., Orlando, FL 32819, 1994.

[2] Pankaj Jalote. *An integrated approach to software engineering (3rd Edition)*. Springer-Verlag New York, Inc., New York, NY, USA, 2005.

[3] Mikael Lindvall, Victor R. Basili, Barry W. Boehm, Patricia Costa, Kathleen Dangle, Forrest Shull, Roseanne Tesoriero, Laurie A. Williams, and Marvin V. Zelkowitz. Empirical Findings in Agile Methods. In *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002*, pages 197–207, London, UK, 2002. Springer-Verlag.

[4] Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[5] FLEXI ITEA2 Project. http://www.flexi-itea2.org/.

[6] C. Larman and V.R. Basili. Iterative and incremental developments. a brief history. *Computer*, 36(6):47 – 56, june 2003.

[7] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for Agile Software Development. http://www.agilemanifesto.org/, 2001.

[8] O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, editors. *Structured programming*. Academic Press Ltd., London, UK, 1972.

[9] Ian Sommerville. *Software engineering (6th ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[10] Kent Beck. *Extreme programming explained: embrace change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[11] Thomas Flohr and Thorsten Schneider. Lessons Learned from an XP Experiment with Students: Test-First Needs More Teachings. In Jrgen Münch and Matias Vierimaa, editors, *Product-Focused Software Process Improvement*, volume 4034 of *Lecture Notes in Computer Science*, pages 305–318. Springer Berlin / Heidelberg, 2006.

[12] Barry Boehm and Richard Turner. Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, 22:30–39, 2005.

[13] Adnan Causevic, Iva Krasteva, Rikard Land, A. S. M. Sajeev, and Daniel Sundmark. An Industrial Survey on Software Process Practices, Preferences and Methods. (ISSN 1404-3041 ISRN MDH-MRTC-233/2009-1-SE), March 2009.

[14] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833 – 859, 2008.

[15] Laurie Williams. Agile Software Development Methodologies and Practices. *Advances in Computers*, 80:1–44, 2010.

[16] M.M. Müller and O. Hagner. Experiment about test-first programming. *Software, IEE Proceedings* -, 149(5):131 – 136, October 2002.

[17] Atul Gupta and Pankaj Jalote. An Experimental Evaluation of the Effectiveness and Efficiency of the Test Driven Development. In *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, ESEM '07, pages 285–294, Washington, DC, USA, 2007. IEEE Computer Society.

[18] Sami Kollanus and Ville Isomöttönen. Understanding TDD in academic environment: experiences from two experiments. In *Proceedings of the 8th International Conference on Computing Education Research*, Koli '08, pages 25–31, New York, NY, USA, 2008. ACM.

[19] Boby George and Laurie Williams. A structured experiment of test-driven development. *Information and Software Technology*, 46(5):337 – 342, 2003.

[20] David S. Janzen and Hossein Saiedian. On the Influence of Test-Driven Development on Software Design. *Software Engineering Education and Training, Conference on*, pages 141–148, 2006.

[21] David S. Janzen, Clark S. Turner, and Hossein Saiedian. Empirical software engineering in industry short courses. Software Engineering Education Conference, Proceedings, pages 89–96, 2007.

[22] John Huan Vu, Niklas Frojd, Clay Shenkel-Therolf, and David S. Janzen. Evaluating Test-Driven Development in an Industry-Sponsored Capstone Project. In *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, pages 229–234, Washington, DC, USA, 2009. IEEE Computer Society.

[23] A. Geras, M. Smith, and J. Miller. A Prototype Empirical Evaluation of Test Driven Development. In *Proceedings of the Software Metrics, 10th International Symposium*, pages 405–416, Washington, DC, USA, 2004. IEEE Computer Society.

[24] Liang Huang and Mike Holcombe. Empirical investigation towards the effectiveness of Test First programming. *Inf. Softw. Technol.*, 51:182–194, January 2009.

[25] Hakan Erdogmus, Maurizio Morisio, and Marco Torchiano. On the Effectiveness of the Test-First Approach to Programming. *IEEE Transactions on Software Engineering*, 31:226–237, 2005.

[26] Lech Madeyski. The impact of Test-First programming on branch coverage and mutation score indicator of unit tests: An experiment. *Inf. Softw. Technol.*, 52:169–184, February 2010.

[27] Matthias Müller and Andreas Höfer. The effect of experience on the test-driven development process. *Empirical Software Engineering*, 12:593–615, 2007.

[28] Andreas Höfer and Marc Philipp. An Empirical Study on the TDD Conformance of Novice and Expert Pair Programmers. In Will Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski,

Pekka Abrahamsson, Michele Marchesi, and Frank Maurer, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 31 of *Lecture Notes in Business Information Processing*, pages 33–42. Springer Berlin Heidelberg, 2009.

[29] N. Juristo, A. M. Moreno, and S. Vegas. A Survey on Testing Technique Empirical Studies: How Limited is our Knowledge. In *Proceedings of the 2002 International Symposium on Empirical Software Engineering*, pages 161–, Washington, DC, USA, 2002. IEEE Computer Society.

[30] Nancy Van Schooenderwoert and Ron Morsicato. Taming the Embedded Tiger - Agile Test Techniques for Embedded Software. In *Proceedings of the Agile Development Conference*, pages 120–126, Washington, DC, USA, 2004. IEEE Computer Society.

[31] Richard F. Paige, Howard Chivers, John A. McDermid, and Zoë R. Stephenson. High-integrity extreme programming. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 1518–1523, New York, NY, USA, 2005. ACM.

[32] Eunha Kim, Jongchae Na, and Seokmoon Ryoo. Developing a Test Automation Framework for Agile Development and Testing. In Will Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski, Pekka Abrahamsson, Michele Marchesi, and Frank Maurer, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 31 of *Lecture Notes in Business Information Processing*, pages 8–12. Springer Berlin Heidelberg, 2009.

[33] Susan D. Shaye. Transitioning a Team to Agile Test Methods. In *Proceedings of the Agile 2008*, pages 470–477, Washington, DC, USA, 2008. IEEE Computer Society.

[34] Michael Puleio. How Not to Do Agile Testing. In *AGILE '06: Proceedings of the conference on AGILE 2006*, pages 305–314, Washington, DC, USA, 2006. IEEE Computer Society.

[35] David Talby, Orit Hazzan, Yael Dubinsky, and Arie Keren. Agile Software Testing in a Large-Scale Project. *IEEE Softw.*, 23:30–37, July 2006.

[36] Kay Johansen and Anthony Perkins. Establishing an Agile Testing Team: Our Four Favorite "Mistakes". In *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile*

Methods - XP/Agile Universe 2002, pages 52–59, London, UK, 2002. Springer-Verlag.

[37] Megan Sumrell. From Waterfall to Agile - How does a QA Team Transition? In *AGILE '07: Proceedings of the AGILE 2007*, pages 291–295, Washington, DC, USA, 2007. IEEE Computer Society.

[38] Lisa Crispin and Janet Gregory. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Addison-Wesley Professional, 2009.