Lappeenranta
University of Technology

Jussi Kasurinen

SOFTWARE TEST PROCESS DEVELOPMENT

Thesis for the degree of Doctor of Science (Technology) to be presented with
due permission for public examination and criticism in the Auditorium 1381
at the Lappeenranta University of Technology, Lappeenranta, Finland,
on the 18th of November, 2011, at 12:00.

Acta Universitatis
Lappeenrantaensis 443

Supervisors	Professor Kari Smolander Software Engineering Laboratory Department of Information Technology Lappeenranta University of Technology Finland
	Dr. Ossi Taipale Software Engineering Laboratory Department of Information Technology Lappeenranta University of Technology Finland
Reviewers	Dr. Mika Katara Department of Software Systems Tampere University of Technology Finland
	Associate Professor Robert Feldt Department of Computer Science and Engineering Chalmers University of Technology Sweden
Opponents	Professor Markku Tukiainen School of Computing University of Eastern Finland Finland
	Associate Professor Robert Feldt Department of Computer Science and Engineering Chalmers University of Technology Sweden

ISBN 978-952-265-143-3
ISBN 978-952-265-144-0 (PDF)
ISSN 1456-4491

Lappeenrannan teknillinen yliopisto
Digipaino 2011

Abstract

Jussi Kasurinen
Software test process development
Lappeenranta, 2011
102 p.

Acta Universitatis Lappeenrantaensis 443
Diss. Lappeenranta University of Technology
ISBN 978-952-265-143-3, ISBN 978-952-265-144-0 (PDF).
ISSN 1456-4491

In this thesis, the components important for testing work and organisational test process are identified and analysed. This work focuses on the testing activities in real-life software organisations, identifying the important test process components, observing testing work in practice, and analysing how the organisational test process could be developed.

Software professionals from 14 different software organisations were interviewed to collect data on organisational test process and testing-related factors. Moreover, additional data on organisational aspects was collected with a survey conducted on 31 organisations. This data was further analysed with the Grounded Theory method to identify the important test process components, and to observe how real-life test organisations develop their testing activities.

The results indicate that the test management at the project level is an important factor; the organisations do have sufficient test resources available, but they are not necessarily applied efficiently. In addition, organisations in general are reactive; they develop their process mainly to correct problems, not to enhance their efficiency or output quality. The results of this study allows organisations to have a better understanding of the test processes, and develop towards better practices and a culture of preventing problems, not reacting to them.

Keywords: organisational test process, test process components, test process improvement, test strategy

UDC 004.415.53:004.05:65.011.08

Acknowledgements

In the present climate of economic crisis and uncertainty, I must use this acknowledgement to express how privileged I feel myself for being allowed to work in such a creative and stable environment. As a man whose high school teacher suggested he should focus on applications for polytechnic institutes, I found myself astonished when I was accepted as a student at this fine university. Again, I was pleasantly surprised when I was selected for a position of a research assistant during the last year of Master's Thesis, and felt it was almost magical when my first publication was accepted for presentation at a conference. My first contribution to the science community, a possibility to teach others things I personally had solved by sheer curiosity and interest, truly a chance to change the world. Now, it feels like it was a lifetime ago. Who knows, maybe it was the magical feeling, but it still lingers in the air every time a new piece of work is accepted. What is the value of the science, if it is not based on interest, a questioning in order to understand and a desire to advance technology and ultimately mankind?

I would like to thank my supervisors, Prof. Kari Smolander and Dr. Ossi Taipale for their contribution and help with this dissertation. I would also like to express my gratitude towards my first supervisor, Dr. Uolevi Nikula, for helping me in learning to master the graduate student work in practice.

Other thanks belong to the other co-authors, Prof. Per Runeson, Jari Vanhanen, Leah Riungu and Vesa Kettunen for their part in this project which finally lead to a dissertation. I would also like to thank Riku Luomansuu for his work with the survey.

In addition, the work of the reviewers of this dissertation, Dr. Mika Katara and Prof. Robert Feldt, your feedback and ideas were a valuable tool for finalizing this work.

A word of acknowledgement for my colleagues at the IT department and friends, for your discussions and support, it helped me greatly, both in this work and personally. It also probably delayed it for at least two years, but who is counting?

Finally, I would like to thank my family, father Ossi and mother Sirpa, and also my sister Kaisla for their support in helping me get through this project.

” A witty saying proves nothing.”

François-Marie Arouet

I would also like to acknowledge the financial support of three contributors to this dissertation, Tekes - Finnish Funding Agency for Technology and Innovation, SoSE - Graduate School on Software Systems and Engineering, and LUT Foundation.

Lappeenranta, 3 October, 2011

Jussi Kasurinen

List of publications

I. Kasurinen, J., Taipale, O. and Smolander, K. (2009). "Analysis of Problems in Testing Practices", Proceedings of the 16th Asia-Pacific Software Engineering Conference (APSEC), 1.12.-3.12.2009, Penang, Malaysia. doi: /10.1109/APSEC.2009.17

II. Kasurinen, J., Taipale, O. and Smolander, K. (2010). "Software Test Automation in Practice: Empirical Observations", Advances in Software Engineering, Special Issue on Software Test Automation, Hindawi Publishing Co. doi: 10.1155/2010/620836

III. Kettunen, V., Kasurinen, J., Taipale, O. and Smolander, K. (2010), "A Study of Agility and Testing Processes in Software Organization", Proceedings of the 19th international symposium on Software testing and analysis (ISSTA), 12.-16.7.2010, Trento, Italy, doi: 10.1145/1831708.1831737

IV. Kasurinen, J., Taipale, O. and Smolander, K. (2010). "Test Case Selection and Prioritization: Risk-based or Design-based?", Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 16.-17.9.2010, Bolzano-Bozen, Italy, doi: 10.1145/1852786.1852800

V. Kasurinen, J., Taipale, O. and Smolander, K. (2011), "How Test Organization Adopt New Testing Practices and Methods?", Proceedings of the Testing: Academic & Industrial Conference: Practice and Research Techniques 2011 (TAIC PART) co-located with 4th IEEE International Conference on Software Testing, Verification and Validation (ICST), 25.3.2011, Berlin, Germany, doi: 10.1109/ICSTW.2011.63

VI. Kasurinen, J., Taipale, O., Vanhanen, J. and Smolander, K. (2011), "Exploring Perceived Quality in Software", Proceedings of the Fifth IEEE International Conference on Research Challenges in Information Science (RCIS), May 19-21 2011, Guadeloupe - French West Indies, France, doi: 10.1109/ RCIS.2011.6006823

VII. Kasurinen, J., Runeson, P., Riungu, L. and Smolander, K. (2011), "Self-Assessment Framework for Finding Improvement Objectives with ISO/IEC 29119 Test Standard", Proceedings of the 18th European System & Software Process Improvement and Innovation (EuroSPI) Conference, Roskilde, Denmark, 27.-29.6.2011, doi: 10.1007/978-3-642-22206-1_3.

In this thesis, these publications are referred to as *Publication I, Publication II, Publication III, Publication IV, Publication V, Publication VI and Publication VII.*

Symbols and abbreviations

ANTI	Application Strategies of New Technologies in Industrial Automation Software Testing, Tekes project
BSI	British Standards Index
CMM	Capability Maturity Model
CMMi	Capability Maturity Model integration
EU	European Union
ICT	Information and Communications Technology
IEC	International Electrotechnical Commission
ISEB	Information Systems Examination Board
ISO	International Organization for Standardization
ISO/IEC 12207	Software life cycle processes
ISO/IEC 15504	Process assessment standard, SPICE
ISO/IEC 25010	Software engineering –Software Product Quality Requirements and Evaluation standard
ISO/IEC 29119	Software and Systems Engineering— Software Testing standard
ISTQB	International Software Testing Qualifications Board
LUT	Lappeenranta University of Technology
MASTO	Towards Evidence-Based Software Quality: Practices and Assessment, Tekes project

MES	Manufacturing Execution System
OU	Organisation Unit
PL	(Safety) Performance Level
SIL	Safety Integrity Level
SME	Small and Medium Enterprises, see (EC 2003)
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability dEtermination model
STS	Socio-Technical System
SQuaRE	Software product Quality Requirements and Evaluation model, ISO/IEC 25010 model
Tekes	Finnish Funding Agency for Technology and Innovation
TIM	Test Improvement Model
TMMi	Test Maturity Model integration
TPI	Test Process Improvement

Contents

1 Introduction

The software testing process is one of the core processes in software development, as every successful software product is tested in one way or another. However, the testing process often has to operate on limited resources in terms of time, personnel or money (Slaughter et al. 1998). To compensate for lack of resources, the test process can be adjusted to cater to the limitations set by the operating ecosystem; in fact, there are studies which conclude that adequate testing can be achieved with low amount of resources, even as low as 15 percent of the requested resources (Petschenik 1985, Huang and Boehm 2006). On the other hand, it is also plausible to say that software testing can become expensive and wasteful if it is done without any preceding planning. A comprehensive set of the test cases including all possible scenarios and outcomes simply cannot be done when software complexity starts rising (Myers 2004). Finally, there is room for developing test process, if only to steer the testing practices towards better efficiency and effectiveness (Bertolino 2007). Observing the software testing from the viewpoint of loss of investment, it is easy to understand why organisations should pay attention to testing activities. In United States alone, the lack of resources and poor infrastructure in testing has been estimated to cause 21.2 billion dollars worth of losses to the software developers. Combined with the losses caused to the clients and customers, this estimate rises to 59.5 billion dollars, from which 22.2 could be saved by making reasonable investments on software testing (Tassey 2002).

The incentive to develop software testing and software quality has been addressed in the development of software industry standards. The new standards, ISO/IEC 29119 (ISO/IEC 2010) for software testing and ISO/IEC 25010 (ISO/IEC 2009) for quality define the testing processes and software quality characteristics. The ISO/IEC 29119 introduces three layers of testing activities; organisational process, divided to test policy and test strategy, test management process and testing work itself, consisting static and dynamic test processes. In this thesis, my research focuses on testing from the organisational viewpoint. From this viewpoint, this thesis explores the concepts presented in the test policies and strategies, such as available test resources, test process activities, test management and quality aspects, basically the whole organisational framework for doing the testing work. This study aims to answer to a research problem "what components affect the software testing strategy and how should they be addressed in the development of test process". This problem is approached from several viewpoints; how do different testing-related components affect the company test process, how can the components defined in the test strategy be used in the development of test process and finally, what concepts should the company address in process development. Additionally, this thesis also discusses the state of testing in software-producing organisations and possible application of ISO/IEC 29119 testing standard to the benefit of actual testing processes in different types of organisations.

For this thesis, both quantitative and qualitative methods were applied and the empirical results were triangulated to improve the validity of the thesis. Our selection of observed level in organisations was in organisational units (OUs) as described in ISO/IEC 15504 (ISO/IEC 2002) to enable us to compare different sizes and types of software companies and make observations on their test processes as a whole. Overall, the high abstraction level constructs were used because using detailed level constructs might have led to too complicated description of the software development process and testing strategies. According to the results of the preliminary studies and existing models such as TMMi2 (TMMi 2010) or ISTQB (ISTQB 2007), the affecting factors and their relationships were analysed from the viewpoint of test process improvement and testing strategy development. Describing the practice of software testing at a high abstraction level was important because, for example, comparing methods, tools and techniques of software testing has a high contextual relevance, and direct comparison between different types of organisations is not feasible approach for scientific, unbiased and universal observation and measurement.

The thesis is divided into two parts, an introduction and an appendix including seven scientific publications. In the introduction, the research area, the research problem, and the applied research methods are introduced, and overall results are presented and discussed. The appendix contains seven publications, which describe the research results in detail. The publications selected for the appendix have gone through rigorous scientific referee process in respected and appropriate publication channels in the software engineering discipline.

The first part, the introduction, contains six chapters. Chapter 2 introduces software testing, viewpoints of the thesis, and the applied testing-related standards. Chapter 3 describes the research problem and subject, the selection of the research methods, and the research process. In Chapter 4, the included publications are summarised. Chapter 5 combines the implications of this thesis for the practice and research. Finally, Chapter 6 summarises the entire thesis, lists its contributions, identifies any possible limitations of the application and suggests topics for further research.

2 Software testing and the viewpoints of the thesis

In this chapter, the central themes and concepts of the thesis are discussed and explained to form a background for the research. The intention is to connect the study to the appropriate context, and explain the viewpoints used in the research subject. The definition of software test process used in this thesis was adopted from the draft of the international standard ISO/IEC 29119 Software Testing Standard (ISO/IEC 2010). According to the standard, software testing consists of three different layers, all of which contribute to the software test process. By researching test processes, the answer was sought to three questions: Which components affect the software testing in practice, what are the important factors from the viewpoint of the test strategy, and how should they be addressed in the development of the test process? In general, what affects the strategy and what concerns should the strategy address.

The research problem can be evaluated from different perspectives, as the process is a compilation of different components and factors, combining technical infrastructure and human interactions to a larger socio-technical (Geels 2004) phenomenon. The research work started with the selection of the viewpoints for this thesis. Test process improvement and testing strategy development were selected as the viewpoints according to the results of the preliminary studies and literature review. This selection was made so as to observe the existing testing process practices from the point of view of software designers, project managers and software testers. This selection enabled us to concentrate research resources on the issues that respondents evaluated as important, and observe the entire testing process, rather than focus on individual mechanisms or process phase activities.

2.1 What is software testing?

The literature contains many definitions of software testing. In the joint ISO/IEC and IEEE standard, a glossary of software engineering terminology, ISO/IEC/IEEE 24765-2010 (ISO/IEC/IEEE 2010), testing is defined as:

> (1) activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation.3.1.46 (IEEE 2008).

The preparation actions, actual testing work and test reporting done in a software project formulates a test process. For example, in ISTQB Glossary (ISTQB 2007) of used terms used in software engineering, the software process is defined as follows:

> **test process:** The fundamental test process comprises test planning and control, test analysis and design, test implementation and execution, evaluating exit criteria and reporting, and test closure activities.

Further, the working draft of the ISO/IEC 29119 standard (ISO/IEC 2010) specifies three layers of testing process, dividing the process of conducting testing to following components:

(1) **Organisational test process,** including test policy and test strategy

(2) **Testing management processes**, including test planning, test monitoring and control and test completion.

(3) **Fundamental test processes**, are further divided into static test processes, which constitute universal activities done with all test cases such as test reporting or case design, and dynamic test processes, which constitute changing activities, such as configuring of different tools or executing a test case.

Related to these layers are the four different concepts of test process, which are defined in the ISO/IEC 29119 glossary as follows:

> **test policy:** A high level document describing the principles, approach and major objectives of the organisation regarding testing.
>
> **test strategy:** A high-level description of the test levels to be performed and the testing within those levels for an organisation or programme (one or more projects).

test management: The planning, estimating, monitoring and control of test activities, typically carried out by a test manager.

test execution:

(1) The process of running a test on the component or system under test, producing actual result(s).

(2) processing of a test case suite by the software under test, producing an outcome (BSI 1998).

(3) act of performing one or more test cases (ISO/IEC/IEEE 24765, Systems and Software Engineering Vocabulary (ISO/IEC/IEEE 2010))

Finally, Kaner and Bach present a shorter definition (2005) as follows:

A technical investigation of the product under test conducted to provide stakeholders with quality-related information.

And in more technical sense in the "Art of Software Testing, 2nd edition" by Myers (2004) as follows:

Testing is the process of executing a program with the intent of finding errors.

Basically, software testing should be defined in these ways because it offers a broad viewpoint on software development. By defining the testing work this way, different approaches to testing tools, development models, resource availabilities and organisation models could be accounted for. However, there is an argument that this definition does not take into account the design-based shortcomings, where the product is working correctly, but the product itself is not correct.

In a traditional sense of testing, the product definition and design are architectural decisions made prior to the software test process. In this mind-set the testing work itself is divided to different types of testing work. In unit testing one module from the software system is tested to validate and verify its functionalities, and it is followed by integration tests, where increasing number of these modules are tested together. Final stages are the system testing, where the entire system is in working condition and tested together and acceptance testing, where customer ensures that the software item is working as intended (for example Behforooz and Hudson 1996). The other concepts related to testing, such as test automation, usability testing or standard compliance, are tools or form the objectives of these test types. This mind-set is also called *incremental testing* by Kaner et al. (1999).

However, in the ISO/IEC 29119 model, all of the verification and validation activities done during the software process are considered to belong to the test process. Validation - confirming that the software is able to fulfil the intended use (ISO/IEC/IEEE 2010) - and verification - confirming that the software complies with the given requirements (ISO/IEC/IEEE 2010) - are both related to the objectives of the test process as defined in the test policy, and exit criteria as defined in the test strategy. Based on the ISO/IEC 29119 standard, the test process should not be understood solely as the roadmap for the traditional development phase where the objective is to find errors with different incremental test types, but in a larger organisational context, including all of the development activities needed to verify and validate the item in testing. By this definition, the test process in the ISO/IEC 29119 asserts all the activities, which are needed to ensure software correctness and design feasibility, namely the validation and verification activities, from first design draft to product release and maintenance throughout the software life-cycle (for example Pfleeger and Atlee 2006).

2.2 What are the test process components?

In this study, the test process is observed and analysed from the perspective of the organisational process. One of the main themes of the study is to understand which test process components have influence on the practical testing work. In this work, the test process component is defined based on the principles of ISO/IEC 24765, in which one of the definitions of a **component** is as follows:

> 2. one of the parts that make up a system. IEEE Std 829-2008 IEEE Standard (IEEE 2008)

In the same standard, a **process** is defined as follows:

> 7. System of activities, which use resources to transform inputs into outputs. ISO/IEC 25000:2005, Software Engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE.4.41 (ISO/IEC 2005)
>
> NOTE [ISO 9000:2005] The term ""activities"" covers use of resources. A process may have multiple starting points and multiple end points. The prescribed manner may be a partially ordered sequence. A process specification can be a workflow specification. An enterprise specification may define types of processes and may define process templates.

The test process components are understood as a group of concepts, which constitute all of the items of the test process, such as test personnel, test tools, test methods, test management, or other. As one of the themes of this study is to identify the important test process components, these concepts are not limited to categories such as technical or social aspects, but used as an umbrella term for every concept, item or activity that has influence on the test organisation or testing work in practice.

2.3 Testing research in general

In software development, the basic objectives of software process are to produce software, which fulfils the required functionalities, has acceptable quality, is completed within budget, and released in time (Kaner et al. 1999). These attributes are all important to the software end-product, as if any of these four – functionality, quality, money and timing – is handled poorly the software is more likely to fail economically. However, in the real world the software development is usually a tradeoff between these four project attributes (Kaner et al. 1999). From this standpoint, it is not very surprising that the testing research is used to develop practices towards better coverage of testing to find more errors, or make the testing work cheaper and quicker while maintaining the pre-existing quality.

Bertolino (2007) lists four desired objectives for the software testing research to pursue: efficiency-maximized test engineering, 100% automatic testing, test-based modelling, and universal test theory. The efficiency-maximised test engineering would mean that the test process could be run on maximum efficiency and effectiveness with the help of smart tools and efficiency-optimised testing methods to ensure good quality (Harrold 2000). The second desired objective, the fully automated testing, aims to build an advanced test automation system, which would be able to do complete autonomous testing work. However, this objective is unlikely to be achieved as even with high degree of automation the system would still need human interaction to confirm results or at least configure and maintain the system (Bach 1997). The third vision of test-based modelling aims at developing software systems towards modelling practices which allow easier and more comprehensive support for testability. The difference between test-based modelling and model-based testing (for example Utting and Legeard 2007) is in the premises; model-based testing tests the software using the model, test-based modelling builds the models based on testability. The last requisite of the universal test theory aims at developing a comprehensive, coherent, and rigorous framework for assessing and comparing the strengths and weaknesses of different testing approaches. The desired objectives of Bertolino may not be very realistic to achieve in the short term, but they all aim at one objective, making testing easier.

The impact of software engineering research in software configuration management is discussed in an article by Estublier et al. (2005). In this discipline of software engineering, the impact of academic studies has been studied in relation to the software industry. Based on the results, it seems that software engineering research and software industry have close relationship; the fundamental systems and new concepts stem from the academia, while industry affects to the development of new technologies. However, as observed by the Estublier et al., the industry may sometimes take several years, even decades to adopt and fully implement the studied concepts. Against this background the current state-of-the-art software engineering and testing research may be still a completely new concept for a real-life software organisation. In fact, in a study by Juristo et al. (2004) it was concluded that even though testing techniques have been studied for over 25 years, there are still several areas that should be examined in more details. Their conclusion is that the testing technique knowledge is still limited, and that over half of the existing studies are based on impressions and perceptions, not on formal foundations, that would allow replicable results. Bertolino (2007) concludes that one way to create the foundation for building test theory is to produce an empirical body of knowledge to understand which factors can explain where the problems arise.

In software engineering and testing research, the empirical studies should be applied to study real-world phenomenon, as the software engineering systems are amongst the most complex systems ever created (Sjøberg et al. 2007). As for the recent development in empirical software engineering and testing studies, Sjøberg et al. (2007) collected metrics on the application of the empirical approach to software engineering publications. They report that between the years 1993 and 2002 approximately 1,800 empirical studies on software engineering were reported in a scientific venue, on average, twenty percent of the published articles. According to Sjøberg et al. (2005), the most common themes of software engineering studies during this period were related to methods and techniques, tools, inter-computer communication, software life-cycles, and product quality.

Juristo and Moreno (2001) discuss the empirical software engineering. The application of knowledge in software engineering discipline is not as straightforward as in other fields. For example, by applying one method to one type of project the results may vary greatly. In software engineering, the basis of acquiring knowledge is iterative; the hypothesis are founded on existing knowledge, but during the observations and data collection in the real world the original hypothesis changes. This process is defined in three steps by Pfleeger (1999):

> Reach an initial understanding, including identifying likely variables, capturing magnitudes of problems and variables, documenting behaviours and generating theories to explain perceived behaviours.

Test theories by matching theory with practice, eliminate variables and identify new ones, determine the relative importance of variables, and identify the range of variable values and probabilities.

Reduce uncertainty, but not always by determining cause and effect.

The important part is to continually question and improve the theory until it explains the observed phenomenon. Some of the measurements may be fuzzy or inaccurate, and some theories may only explain the phenomenon partially. However, it is better to have a partial understanding that can serve as a basis for future theory than to discard a result simply because it does not explain or cover everything (Pfleeger 1999).

2.4 Testing as defined in the ISO/IEC 29119 Software Testing standard

The working draft of the upcoming ISO/IEC 29119 Software Testing standard considers the test process and the testing work to include both the activities to validate and verify the item being tested. Unlike earlier testing-related standards such as IEEE 829 (IEEE 2008) which outlines a standard for content of different test documentation, or IEEE 1008 (IEEE 1987), which defines unit testing as an activity to plan, create and use a set of test items, ISO/IEC 29119 aims to combine the concepts of the earlier and more specific standards to one test process model encompassing the entire software organisation. In the ISO/IEC 29119 standard, the test process encompasses the entire organisation, beginning from the upper management, policies, and quality requirements. The organisational policies and strategy steer the testing work at the project level, where the project level management creates test plans, and monitors and controls the testing activities at the fundamental level. Based on the fundamental level results, a test completion report is created and used along with the feedback to develop testing at both the project and organisational level. The process model is illustrated in more detail in Figure 1.

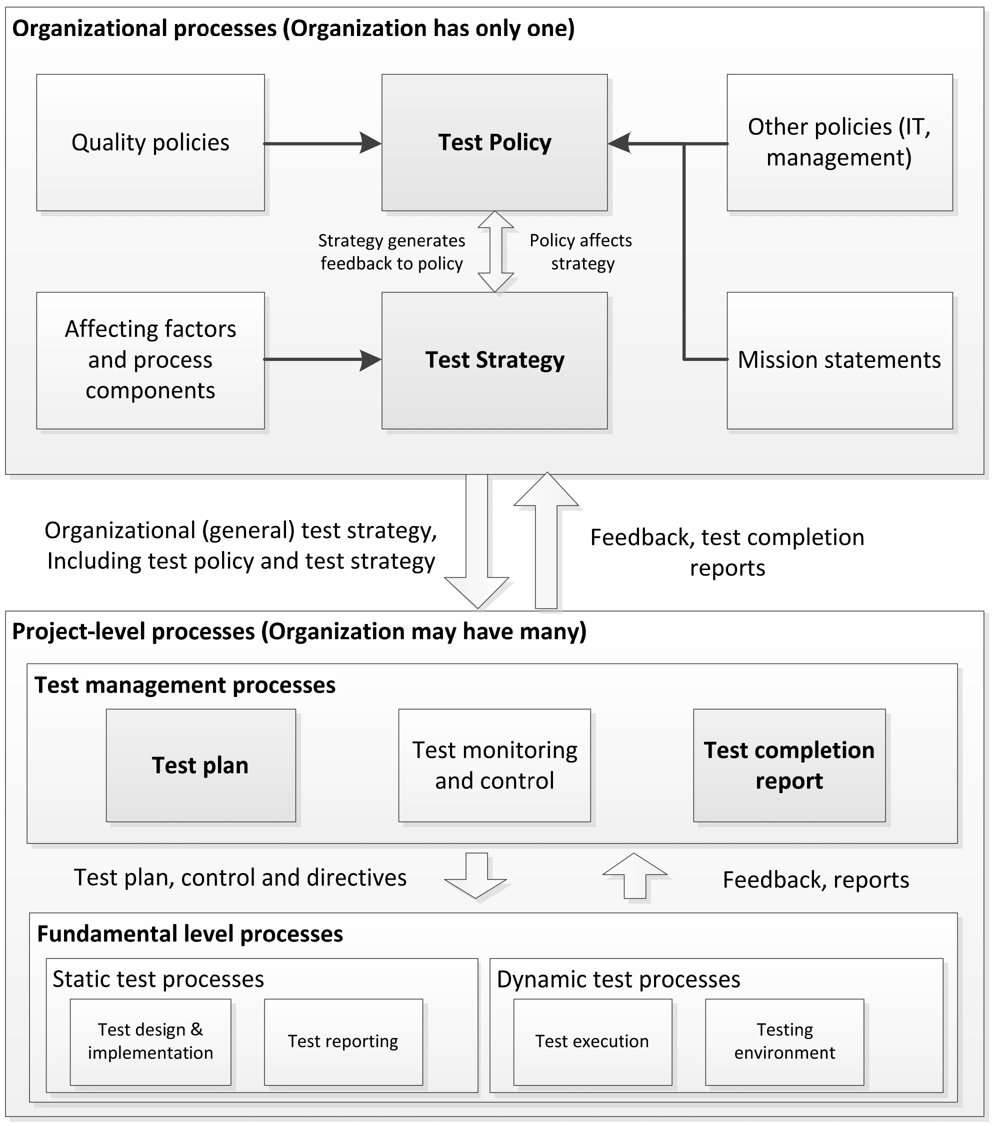

Organizational processes (Organization has only one)
Quality policies
Test Policy
Other policies (IT, management)
Strategy generates feedback to policy
Policy affects strategy
Affecting factors and process components
Test Strategy
Mission statements
Organizational (general) test strategy, Including test policy and test strategy
Feedback, test completion reports
Project-level processes (Organization may have many)
Test management processes
Test plan
Test monitoring and control
Test completion report
Test plan, control and directives
Feedback, reports
Fundamental level processes
Static test processes
Test design & implementation
Test reporting
Dynamic test processes
Test execution
Testing environment

Figure 1. ISO/IEC 29119 Test process test levels and structure

The ISO/IEC 29119 testing standard defines four main documents for the test process, which define how the organisation and individual projects should perform testing. These documents are test policy, test strategy, test plan and test completion report. Test policy and test strategy are organisational level documents; the organisational management defines these documents to steer the test process in the project level. At the project level, project management is responsible for defining the test plans for the project based on the organisational level policies and strategies. Project-level management is also responsible for reporting feedback from the project to the organisational management by compiling test completion reports, which then are assessed and form a basis for the organisational test process improvement. The process produces four documents; test policy, test strategy, test plan and test completion reports.

Test policy is a short, two-page document defining the test process on a highly abstract level. Test policy defines the scope, principles and rules for testing to which the organisation should adhere. The main concept is that the test policy defines what is accomplished with testing, leaving the actual implementation for the other documents. Based on the standard, the following items should be addressed in a test policy (ISO/IEC 2010):

Objectives of testing describing the purpose, goals, and overall scope of the testing within the organisation.

Test process states the test process that the organisation follows.

Test organisation structure states the titles and positions of individuals who belong to the test organisation and a diagram to illustrate the organisational hierarchies.

Tester education states the required certifications and education required for members of the test organisation.

Tester ethics state the ethics code to be upheld by the test organisation and testers.

Standards define the standards the test process activities are required to follow.

Test asset archiving and reuse strategy describes how test assets should be archived and how the assets are to be reused.

Test process improvement states the methods for measuring and improving test process in the future.

Measurement for value of testing defines how return of investment from testing should be calculated.

Other relevant policies are also listed and defined in case the organisation has overlapping or important policies beyond testing activities, such as quality policy.

In addition to the test policy, the organisational management also defines test strategy, which is a more specific and detailed document describing how test activities should be done in the projects. According to the ISO/IEC 29119 standard, a test strategy should address the following items:

Generic risk management should identify generic risks that may impact testing activities and offer methods to reduce these risks.

Entry and exit criteria should be defined to determine when testing should be started and what the criteria are for ending test activities for an item under test.

Degree of independence should be established to define how technically, managerially and financially independent the test organisation is from the other parts of the organisation.

Test organisation structure should be defined in the test strategy if the test organisation structure differs from the one defined in the test policy.

Test documentation strategy identifies the test documentation used in the testing work.

Test phases, which phases are performed during testing, should be identified and defined.

Test types, which types of testing should be performed as a part of the test process.

Test techniques, which specific test techniques should be used in the testing work should be identified.

Test selection and prioritization method for selecting applied test cases and their internal prioritization should be defined.

Test environment strategy identifies the testing environments; where test should be performed, who is responsible for the environment, where the test data is located, who is responsible for test data and how the test data is acquired.

Test automation and tools defines the organisational approach on test automation, and identifies the test automation tools and their application in the test process.

Retesting and regression testing should be described to establish the strategy and conditions for retesting and regression testing.

Test completion criteria should be defined to establish a clear criteria for conditions where the testing activities should be considered completed.

Incident management strategy should be described to establish how incidents should be managed during testing activities.

The level of details in the test strategy is more refined than in test policy, and may include clear indicators and thresholds to steer test process at the project-level. At the project level, the test policy and strategy are applied as a foundation, when a new test process is being defined. At the project-level, the management defines a third test process definition, a test plan, based on the principles and criterion set by the organisational documents. The test plan further elaborates on the same topics as strategy, and should include the following items (ISO/IEC 2010):

Project and the processes and products the plan will be used for should be defined.

Test item(s) should be identified to define which items, such as complete software, interface between units or subsystem are covered in this test plan, and describe the purpose of the system and possible references for more information.

Test scope should define the high-level list of included and excluded parts of the item in testing. The scope should also clearly specify the limits of the test effort and identify systems, which are specifically excluded from the testing.

Test strategy should define the strategy that will be applied in this test plan, including test phases, test types, test design techniques, test environment, test tools, and deliverables from the testing work.

Estimates of the test tasks should describe the work break down of the testing work within the scope of the test plan, identify the milestones, and offer estimates for the test budget and cost estimates.

Staffing should be defined to establish tasks and responsibilities, hiring needs and possible training needs for the test organisation.

A schedule should be presented to summarise the overall schedule of the testing tasks and different test phases and different support processes.

Cross-reference to risks should depict the relationship between project risks and how the different test activities acknowledge these risks.

Deviations from the organisational test strategy should also be separately listed in cases where the test plan does not follow organisational strategy, and identify the authorities which have approved these deviations.

In addition to designing test plan, the project-level management is also responsible for providing feedback to the organisation from the completed test processes. In the standard model, the project-level management can achieve this by compiling a test completion report. This report summarises the testing work done during the project, and lists the deviations, collected metrics, and reusable assets from the test process. The test completion report should include the following information (ISO/IEC 2010):

Summary of testing performed summarising the testing performed in the different test phases and the different test types.

Differences from planned testing should also be listed in the report to provide information on the deviations from the plans.

Test completion criteria should be described and specified on how the testing work met the set criteria, or why the testing was unable to reach the set criteria.

Factors that blocked the process should be identified and described with enough details to enable the organisation to remove them in future projects.

Lessons learned should document the results of the lesson learned (post-mortem) meeting.

Test metrics should provide information on collected test metrics such as amount of test cases, found defects and incidents.

New and changed risks should list the newly identified and changed risks along with the taken actions to prevent these risks.

Test environment status should describe the final state of the test environment after the testing was completed.

Test deliverables should list and reference the produced deliverables from the project.

Reusable test assets should be listed for possible future projects, along with the information regarding the location and availability of said assets.

Recommendations should define the recommended use of the test item based on the results of the completed testing work.

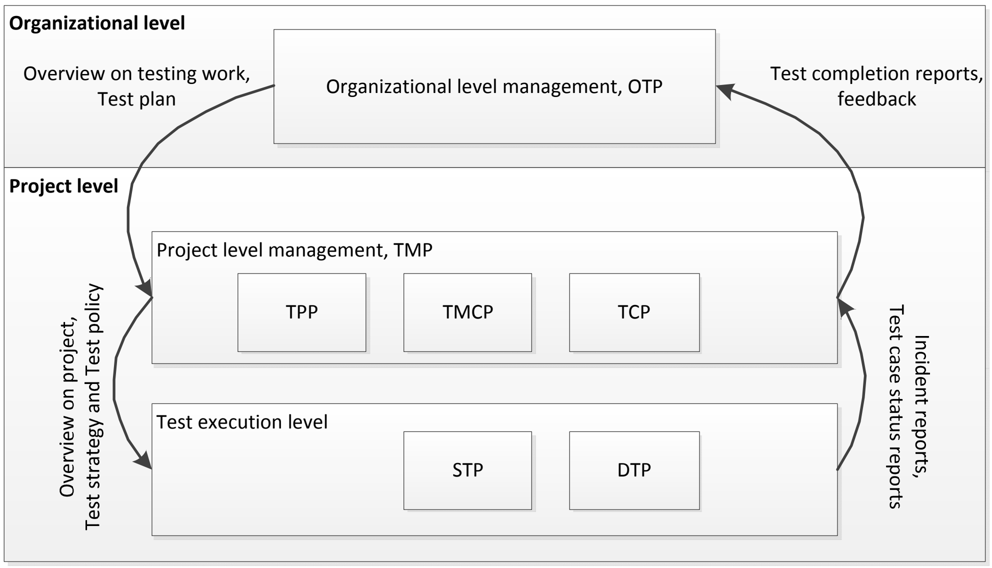
These are the four main documents, which are used in design and improvement of test process in the organisation. The standard also defines other document types, such as test status reports and rules for test design, but they are more related to everyday management and testing work steering activities than defining the actual test process.

Besides documentation, the standard process model is layered into three levels, which are (1) organisational test processes, (2) test management processes in project-level and (3) fundamental level, which constitutes (a) static and (b) dynamic test processes. In these layers, the testing activities are further divided into sub processes, which define the different activities happening in the layers. These processes are as follows (ISO/IEC 2010):

> **The Organisational test process** (OTP) is used to develop and manage organisational test specifications, such as test policy and test strategy. It is also responsible for monitoring and controlling that testing activities use the organisational level specification.
>
> **Test management processes** (TMP) are the project-level management activities in the test process. TMP defines the test planning, test monitoring and control and test completion. They are also responsible for updating the test plan at the project-level.
>
> **The Test planning process** (TPP) is the process which is responsible for developing the test plan. Depending on the project phase, this may be a project test plan, or a test plan for a specific phase such as system testing or acceptance testing.
>
> **Test monitoring and control process** (TMCP) ensures that the testing is performed in line with the test plan and organisational test documents. It is also responsible for identifying updates necessary for the test plan.
>
> **The Test completion process** (TCP) is a process that includes activities, which are done when testing is completed. It ensures that useful test assets are made available for later use.
>
> **Static test processes** (STP) describes how static testing activities, such as test preparation, test result reviews and analysis and test follow-up are done. These activities are the "general" activities, which are done to all test cases in all test phases of the project, such as reserving test resources, reviewing the test results and seeing through that necessary follow-up actions are done based on results.
>
> **Dynamic test processes** (DTP) describe how dynamic test activities such as test implementation, test execution, test environment set-up, and test incident

reporting are done in the organisation. These activities are the "practical" activities, which vary between different types of testing, including configuring test tools, deciding test conditions based on test documents and practical tasks of preparing test cases and test sets.

In the ISO/IEC 29119 standard, some of these processes, such as STP or DTP, are also divided into smaller sub-categories within these definitions. This does not affect the overall meaning of the process, but rather further illustrates and explains the purposes of the activities they represent. Some processes, such as TMP, are also the owners of the other processes of the standard. The relationships between the model processes are illustrated in the Figure 2.



Organizational level
Overview on testing work, Test plan
Organizational level management, OTP
Test completion reports, feedback
Project level
Project level management, TMP
TPP
TMCP
TCP
Overview on project, Test strategy and Test policy
Incident reports, Test case status reports
Test execution level
STP
DTP

Figure 2. ISO/IEC 29119 Test process processes divided into different test levels

The central theme of the ISO/IEC 29119 standard is the division of operating into two main levels; organisational management, and individual projects. Even if the process model is detailed, it should be adjustable to several different software domains. It is obvious that control software of a jet fighter will probably be built differently than a mobile game, but the process model aims to allow different approaches within the same overall process model. This is achieved by adjusting the organisational documents, which define the framework for the project-level work.

2.5 The viewpoints of this thesis

The effects of different test process components and test process development were selected to be the viewpoints of this thesis in order to understand how test strategy can be defined and where the organisations should focus their process improvement effort. The scientific objective was to study how different test process components affect the practical testing work, and how the test organisations could be developed based on the principles and practices presented in the ISO/IEC 29119 test process standard. In the following segment, some of the most interesting test process components with possible influence on the test process activities are discussed, followed by a segment briefly introducing the test process improvement and its concepts in practice.

2.5.1 Test process components

In software testing, the test strategy encompasses several components, which have a direct effect on the testing activities. The test strategy is the core of the test process; it defines the test process concepts by setting an overall framework for testing: the objectives and defining methods and resources available to the test work in the lower layers of the model. The strategy is a high-level document, which has a large influence on several test process components, as illustrated in the Figure 3. In Figure 3, the different components which are identified by different test certificates (TMMi2 2010, ISTQB 2010) and the upcoming ISO/IEC 29119 standard are collected and loosely categorised into five categories. The sixth category "Possible areas of interest" is then taken from the concepts suggested by the other sources, such as existing research literature and previous research results from the ANTI project. The figure also divides the components into the dissertation viewpoints; on the right hand side, the components of interest, which define the organisational test strategy are listed, while the left hand side constitutes the different levels of test process activities which constitutes the organisational test process.

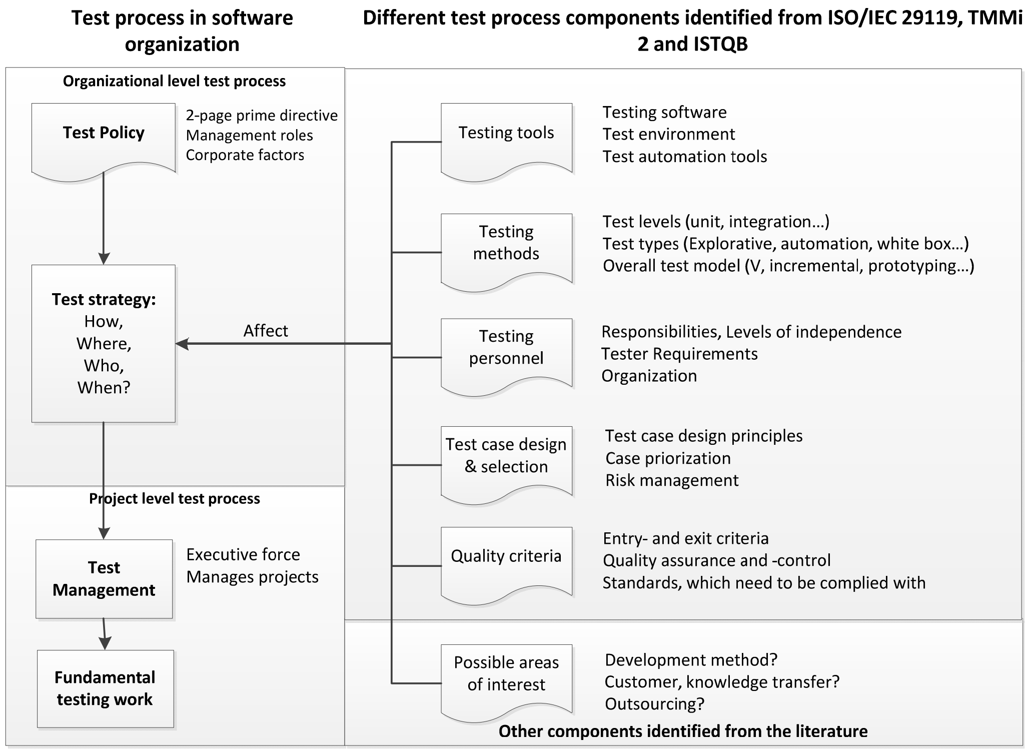
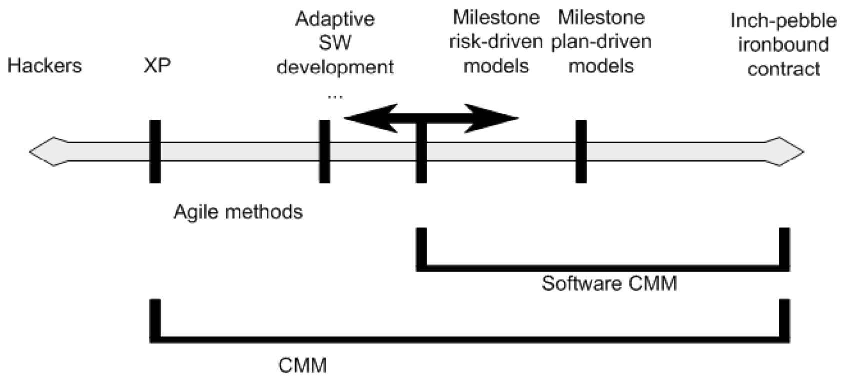

Test process in software organization
Different test process components identified from ISO/IEC 29119, TMMi 2 and ISTQB
Organizational level test process
Test Policy
2-page prime directive
Management roles
Corporate factors
Test strategy:
How,
Where,
Who,
When?
Affect
Project level test process
Test Management
Executive force
Manages projects
Fundamental testing work
Testing tools
Testing software
Test environment
Test automation tools
Testing methods
Test levels (unit, integration…)
Test types (Explorative, automation, white box…)
Overall test model (V, incremental, prototyping…)
Testing personnel
Responsibilities, Levels of independence
Tester Requirements
Organization
Test case design & selection
Test case design principles
Case priorization
Risk management
Quality criteria
Entry- and exit criteria
Quality assurance and -control
Standards, which need to be complied with
Possible areas of interest
Development method?
Customer, knowledge transfer?
Outsourcing?
Other components identified from the literature

Figure 3. Different test process components and the levels of the ISO/IEC 29119 model in the test process of the software organisation

In the test strategy, the organisation defines several components for the test process, which all affect the testing work, such as testing tools, available time and testing personnel. It has been established, that the lack of investment in the testing infrastructure causes losses worth several billion dollars (Tassey 2002), but the studies also indicate that improving the testing infrastructure is not cheap or easy to implement. In a study by Ng et al. (2004), the most common barrier on adoption of new testing tools were considered to be the costs associated with the adoption process, the time consumption the adoption process takes and the difficulty of adopting new tools. Similarly, on adoption of testing methodologies, lack of expertise was considered the most important reason preventing the adoption of new test methodologies, and providing training was seen as too costly and time-consuming to allow investment in a real software-producing organisation.

In the traditional software development models such as the waterfall model (for example Pfleeger and Atlee 2006), the testing work usually follows the main development phase of the software. In this approach, the testing phase should not include changes to the design or requirements, but in reality, the software may still undergo changes, especially if the customer has influence on the development (Highsmith and Cockburn 2001). To address this issue, a new trend regarding the software development approach, agile development, has been introduced (for examply Abrahamsson et al. 2002). In a publication by Abrahamsson et al. (2002), agile methods are described as an attempt to answer to the business community asking for lighter-weight and more adaptable software development process. The agile models differs from traditional, plan-driven, development models by promoting communication between stakeholders and production of working releases instead of excess documentation and design before implementation (Fowler and Highsmith 2001). In comparison between plan-driven approaches and agile methods, the main difference can be characterised to be the amount of planning, as illustrated in Figure 4 by Boehm (2002).



Hackers
XP
Adaptive SW development ...
Milestone risk-driven models
Milestone plan-driven models
Inch-pebble ironbound contract
Agile methods
Software CMM
CMM

Figure 4. Development methods using a planning spectrum (Boehm 2002)

In the planning spectrum, the agile methods and traditional methods are defined based on the amount of planning in the development. By this definition, the different development methods fall between two polar stereotypes; hackers, who make only little to no plans before implementing code, and ironbound contracts, where every change or addition to the software has to be agreed upon and documented. Overall, the implication is that in plan-driven development, the process prepares for possible hindrances, whereas in agile development, the process reacts to the issues that arise.

However, this model is a somewhat idealistic view on agile development. In practice, not all development processes are possible to convert to agile practices simply by making changes in the planning activities. For example, the size of the project or criticality (Boehm and Turner 2003) of the end-product may steer the development process towards a plan-driven approach. To express these considerations, a more detailed model for general feasibility of agility in the development process has been developed by Boehm and Turner (2003). Figure 5 illustrates this model. Based on the model, the organisations that are on the outside rim of the model are more likely to encounter difficulties in application of agile practices. However, this does not mean that the agile practices are impossible to implement in large organisations; there are cases, where large organisations have applied agile practices such as SCRUM (Deemer et al. 2010) in their development process.

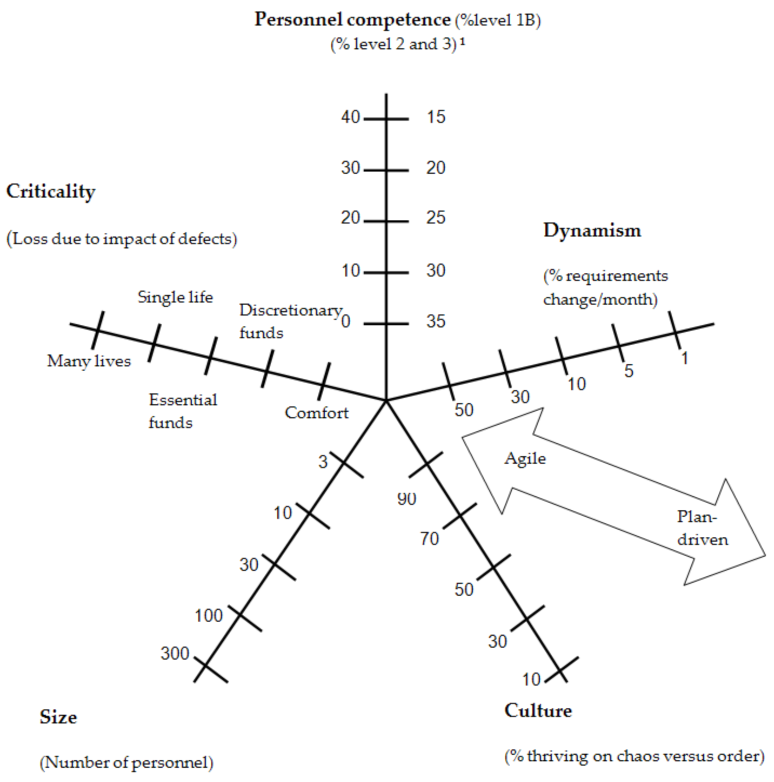


Figure 5. Agile versus plan-driven methods (Boehm and Turner 2003)

Software testing aims to improve the quality of a software product, and in fact is a major component on deciding if the software project is profitable (Huang and Boehm 2006). However, in the measurement of quality, the definition of quality can be troublesome, as the concept of quality is closely related to a number of subjective observations. For example, Garvin (1984) has discussed the definitions of quality and made extensive definition work for establishing what the quality actually is and how it affects product concepts such as profitability or market situation. Garvin defines five different definitions for quality; transcendent, product-based, user-based, manufacturing-based and value-based definition. Even though they define the same phenomena, product quality, they vary greatly. For example, transcendent quality is "innate excellence", which is absolute and uncompromising standard for high achievement, certainly identified if present. On the other hand, user-based quality is the more common "satisfies user needs"-definition, whereas manufacturing-based definition promotes conformance to the product requirements. Garvin also discusses the different definitions by mentioning that it also explains why different people seem to have differing opinions as to what constitutes quality; they tend to apply the definition they are most familiar with.

The different aspects and definitions of quality also mean that the measurement of software quality has some considerations. A paper by Jørgensen (1999) introduces three assumptions for establishing measurement for software quality: There are no universal quality measurements, but meaningful measures for particular environments. Secondly, widely accepted quality measurements require maturity in research and thirdly, quality indicators predict or indirectly measure quality. In short, Jørgensen establishes that there are no universal measurements, but the approaches using quality indicators – characteristics and attributes – can be used to approximate or predict software quality.

Jørgensen also discusses the different aspects of software quality. In addition to a set of quality factors, there also exist other definitions for quality; quality as user satisfaction and quality as the degrees of errors in software. However, both of these other models have serious flaws. In quality as user satisfaction, the most obvious flaw lies in the question as to why the measurement of user satisfaction is called software quality? There exist several groups of users for software, such as administrators and basic users, so how can the total satisfaction be calculated? Furthermore, how can the user group A's "very satisfied" be related to group B's "very satisfied"? They may not even mean the same concept, or at least may not be based on the same features. In the quality as the degrees of errors, the problem lies within the classification; how many flaws in the user interface relate to a critical system error? Therefore, by Jørgensen's definition, the most sensible model for estimating quality seems to be based on the characteristics, observing different aspects of the software. However, criticism also exists towards this approach, for example by Salvaneschi and Piazzalunga (2008).

In the ISO/IEC 25010-3 Software product Quality Requirements and Evaluation standard (2009), the definition of software quality is similar to the interpretation presented by Jørgensen. In the standard, the software quality is defined in generally applicable and measurable terms. The quality is presented as a composite of eight quality characteristics, such as operability, security, or compatibility. These characteristics are further divided into sub-characteristics such as fault tolerance, accuracy, or compliance, which aim to be measurable either by internal or external measurements (ISO/IEC 2009). The product quality is understood to be an amalgam of all of the quality characteristics, with a prioritization and weight distribution based on the quality objectives. The quality model is illustrated in further detail in Figure 6.

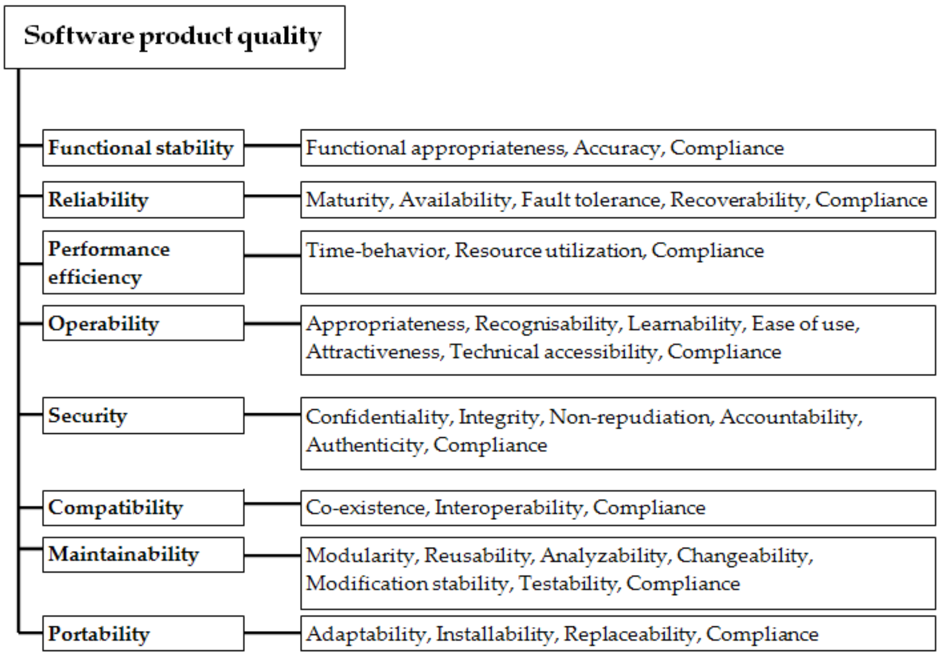

Software product quality
Functional stability
Functional appropriateness, Accuracy, Compliance
Reliability
Maturity, Availability, Fault tolerance, Recoverability, Compliance
Performance efficiency
Time-behavior, Resource utilization, Compliance
Operability
Appropriateness, Recognisability, Learnability, Ease of use, Attractiveness, Technical accessibility, Compliance
Security
Confidentiality, Integrity, Non-repudiation, Accountability, Authenticity, Compliance
Compatibility
Co-existence, Interoperability, Compliance
Maintainability
Modularity, Reusability, Analyzability, Changeability, Modification stability, Testability, Compliance
Portability
Adaptability, Installability, Replaceability, Compliance

Figure 6: Software product quality model as presented in ISO/IEC 25010

In addition to the software quality characteristics, another indicator for software quality requirements is the criticality (adapted from Boehm and Turner 2003, Huang and Boehm 2006). Software criticality is an approximate indicator, indicating the worst possible outcome for the software failure. Unlike other similar measurement such as safety integrity level (SIL, see Brown 2000) or safety performance level (PL, see Soressi 2010) which both measure the probability of failure against hours operated, software criticality is much more simple measurement as it does not require strict metrics or measurements. It simply represents the possible worst-case scenario directly caused by the failure of the software. The criticality is represented as a scale from one to five, with the following descriptions for each level of criticality:

1: None or at most user irritation; for example "user has to reboot the game system"

2: Small economic losses; "the ticket fails to print and money is lost", "no record of sale is made"

3: Significant economic losses; "Store has to be closed for a couple of days", "product stock has to be scrapped".

4: Bodily harm or great economic losses; "Operator loses hand", "production line has to be closed for repairs."

5: Loss of human life; Operator or people depending on the software system are killed.

The criticality of the software product may affect the quality objectives of a software organisation, and possibly correspond with the amount of resources allocated to the test process.

2.5.2 Test process development

The objective for this dissertation was to observe and identify important test components to understand how they should be addressed from the viewpoint of test process development. Identification of the important components could offer external assistance to the organisations in the adoption of practices and operating models such as the model illustrated in the ISO/IEC 29119 standard. This objective required the study to observe the test process and test strategy from a viewpoint that consisted of all test process components, and observations regarding how the real-life organisation developed their testing practices.

The first objective was to assess whether the ISO/IEC 29119 model itself was feasible enough to implement in a real-life organisation. To assess the test process model feasibility, an understanding was required of the software process improvement (SPI) in real-life organisations. SPI literature includes studies about the effect of different factors on software process improvement. For example, a study by Abrahamsson (2001) discusses the requirements for successful process improvements. The most important factor according to the Abrahamsson study is the commitment to change from all organisational levels. If some of the levels disagree with the process improvement, the process improvement process tends to fail. In addition, the process improvement has to be executed in a controlled, well-planned, and organised way to ensure the possibility of permanent, positive improvements. In a more specific example, Pino, Garcia and Piattino (2009) discuss process improvement in small-sized companies. They conclude that process improvement should define management-level commitments immediately after the improvement process is established, and that improvement proposals are sometimes hard to align with the strategic planning in the organisation. They also consider that organisations should have advisers to initially trial the first few improvement iterations. Similar findings are also reported in the articles by Sulayman and Mendes (2010) and Hardgrave and Armstrong (2005). In addition, the article by Wong and Hasan (2008) also includes cultural influences in process improvement considerations. Culture, whether it is organisational culture or national culture, affects the requirements for effective process improvement. For this reasoning, as process improvement would assume cultural changes, it is important to also study the aspects of the social science in SPI (Conradi and Fugetta 2002).

In studies applying certain process models in organisations, the Hardgrave and Armstrong study (2005) observed that their case organisation had trouble reflecting their existing process in the theoretical models. In their paper, the organisation estimated the time needed for process improvements to achieve CMMi (CMMi 2010) level 2 as 10 months, when in fact the entire process took four years. In their reported case, the organisation decided to employ external advisors after 16 months of internal process improvement. Hardgrave and Armstrong also conclude that organisations tend to lose the initial drive for process improvement because the drive for an improvement process, in many cases, is not the internal need to develop, but rather to reach out for certain external rewards, such as certifications. Kautz, Hansen and Thaysen (2000) describe a case, where a simplified iterative development model was introduced into an organisation and applied in the practice. Their main finding was that organisations can adjust to given models, provided that the model itself is sound and is not too strict with the process requirements.

Dybå (2003) conducted a study on SPI activities in different types of organisations. They concluded that the company size does not hinder or restrict the process improvement activities. Small organisations are at least as effective as large ones in implementing process improvement. Small organisations tend to be less formal in organisational hierarchy and in turbulent business environments they use explorative test and development methods (see Kaner et al. 1999) more willingly. Another interesting observation was also that organisations have a tendency to define their own best practice methods, as regards what is working, while failure in process improvement is considered an unacceptable possibility. As process improvement projects often fail, companies tend to support the status quo if corrective actions are not absolutely necessary. Dybå also discusses the explicit process definitions, which should also be understood as a guideline; informal practices are used to supplement the formal way of working, and collect experiences for the subsequent improvement efforts.

Overall, the literature indicates that organisations can adopt different models if the model is sound and reasonably adjustable (Kautz, Hansen and Thaysen 2000), and that the size of organisation does not restrict their ability to make process improvements (Dybå 2003). It is also indicated, that process development is dependent on several stakeholders and other contributing factors (Abrahamsson 2003, Wong and Hasan 2008), and that organisations tend to have difficulties in observing and changing their own processes without external assistance.

In the development of the process improvement framework, the general requirements for any relevant construct should include, at least, that it is acceptable to the software development community and that it is based on agreed software engineering principles and practices (Burnstein et al. 1996). For example, the validity issues for developing frameworks have been addressed in prior studies (Jung 2009, Karlström et al. 2005). Jung (2009) developed a test process maturity model based on internal needs, and validated the results via a case study and a survey. Similarly, with the minimal test process framework (MTPF) developed by Karlström et al. (2011), the initial model was designed based on observations in real-life organisations, and further elaborated and validated with surveys and an empirical case study.

In large-scale process development, there are some testing-related process frameworks for test process development, such as Test Maturity Model (TMM) (Burnstein et al. 1996), Test Process Improvement (TPI) model (Koomen and Pol 1999) and Test Improvement Model (TIM) (Ericsson et al. 1997). The TMM framework was developed from the principles of CMM and a group of other pre-existing practices (Burnstein et al. 1996) to allow organisations to develop their test process towards better principles and practices. TMM was developed with three main objectives. The first objective was to create a set of levels that define testing maturity hierarchy, where each level represented a stage of evolution towards mature testing capacity. The second objective was to create a set of maturity goals for each level, which gives the organisation a concrete example for development. The third objective was to create an assessment model, which would allow the organisation to obtain a clear understanding of their situation (Burnstein et al. 1996). Currently, the TMMi reference model covers 16 test process areas, divided into five different maturity levels from managed process at level 2 to the self-optimizing process at maturity level 5 (TMMi 2010).

Another model for test process development is the Test Process Improvement (TPI) (Koomen and Pol 1999). TPI model is based on the assessment of maturity levels in different key areas such as life-cycle model or metrics, and comparing the existing level of maturity against the set objective level. The assessment model has corner stones, several key areas and levels, which are defined in close detail, and which in turn produce very detailed improvement suggestions. However, like other large process development models, it has a number of limitations, especially in scalability (Farooq and Dumke 2007) and applicability (Jung 2009) in practice.

Test Improvement Model (TIM) development has been based on developing the TMM and CMM (See CMMi 2010) principles further, by introducing the positive traits of the existing process models in a new context and new application method. The TIM development focused on two major components, a framework consisting of level ladders and key areas, and the assessment procedure itself. The important innovation of the TIM model was the ability to assess the current state of the practice in the key areas of testing independently, and put the assessed organisation "on the map" with their current test process.

The viewpoint of this thesis as regards process development and test process improvement is not as straightforward and objective as it may seem, as there are some considerations in the development of a framework for adopting the existing process model. The development of a framework for self-assessment and adoption was necessary, as the test process models (such as the TMM and subsequently the ISO/IEC 29119 model) are rather difficult to adopt in a real-life organisation, as they lack the guidelines in adoption of the process activities, and organisations tend to try to preserve the status quo (Dybå 2003). Organisations also tend to favour only the process improvement proposals, which they can relate to (Hardgrave and Armstrong 2005). Even if the adoption model exists, the adoption process is not easy to implement; for example the TMM process adoption model TMMi (TMMi 2010) has been criticised for being counter-intuitive (Jung 2009) and unrealistic to implement (Oh et al. 2008) even if the model itself is fundamentally based on the best practices and related standards (Burnstein 1996).

2.6 Summary

In this thesis, the software test process is evaluated from the viewpoint of test strategy components and test process improvement. In the software process, the test process and components related to the test process have a large influence on the product outcome. The test process, as defined in the ISO/IEC 29119, goes beyond the traditional concept of a waterfall-type project test phases, where product faults are identified and corrected, to including all validation and verification activities for the software item, during the software lifecycle, in an organisational and project-level context. In the test process, test strategy defines several aspects such as test resources, test planning activities, test management activities, quality criteria and applied test methods, which create a framework for the test work in practice at the project level. By identifying the most influential test process components and their relationship to the whole software process, a framework for test process improvement can be defined to steer the test work in real-life organisations towards better practices such as increased cost-effectiveness or risk-avoiding techniques. Overall, the objective is to enable the organisations to assess their test process needs more accurately, and be able to develop their testing work towards better practices and better quality.

3 Research goal and methodology

In this chapter, the research goal is introduced and the applied research methodology explained. This chapter also discusses the reasoning behind the selection of the applied research approaches, and describes the data collection process.

To approach the research problem, i.e. "what components contribute to the software testing process and how should they be addressed in the development of a test process", the problem was decomposed into a group of sub-problems which were discussed separately in the respective publications. The objective of the first sub-problem was to identify the contributing factors of the thesis from the prior research into the topic. The objective of the other sub-problems was to study the effect and relevance of the identified testing factors and derive process improvement hypotheses by analysing the research subjects from selected viewpoints with quantitative and qualitative methods.

Software testing and related software development in organisations formed the research subject. To initially describe the studies subjects, international standards were used to define a software organisation and the activities and sub-processes which happened within the organisation. The standards ISO/IEC 12207, Software Life Cycle Processes (2008), ISO/IEC 15504-1, Concepts and Vocabulary (ISO/IEC 2002), ISO/IEC 25010-3 Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) - Quality Model (ISO/IEC 2009) and the working draft for ISO/IEC 29119 Software and Systems Engineering— Software Testing (ISO/IEC 2010), define a software organisation, which was used as an a priori framework for the research subjects. The ISO/IEC 29119 test process model was applied in this study even though the international standard was still only a draft as it defined a size and maturity-independent definition of the test process in software organisations. The concept of the standard model is to be applicable in any organisation testing software, regardless of the size, business domain or product type, and simultaneously be applicable in cooperation with other established international standards such as ISO/IEC 12207 and ISO/IEC 15504. In this study, the research subject was initially understood to have processes in both software development and testing; conducting one coherent software process similarly as defined in the standards ISO/IEC 12207 and ISO/IEC 15504 in development and ISO/IEC 29119 in testing.

The research process consisted of three phases: preliminary study (viewpoints of the thesis), main data collection and analysis phase (identification of important factors) and validation phase (studies in test process improvement). In the selection of the research methods, the objective was to find the best method to approach the research subject. For the preliminary phase of the thesis, the Grounded Theory method (Strauss and Corbin 1990) was selected for the analysis of the prior data, a decision which was based on the type of the existing data, and considered feasible approach for extended application in the latter qualitative research phases. The survey method (Fink & Kosecoff 1985) was used for the quantitative phase of the thesis.

3.1 The research problem

The research problem arose from the author's research group's research concerning the software development and testing and a literature review on the existing studies on software testing. The decision to focus on organisational aspects of the test process and in the development of test process in organisation was also in accordance with general research themes of research group's new research project called MASTO, which studied software testing and standardizations in real-life organisations.

According to the literature, more than 50 % of development effort is frequently focused on testing (Kit 1995, Koomen and Pol 1999). On the other hand, testing can also be effective with only a small portion of the "optimal" resources (Petschenik 1985, Huang and Boehm 2006) and in many cases, the test processes have to adjust to resource limitations (Slaughter et al. 1998), so as an organisation the test organisation has to be adaptive and to some degree, even creative. However, the studies on organisational decisions and activities concerning the test strategy composition and test process components are less common. There are some organisation level studies which introduce organisational level test process components (Ng et al. 2004) and metrics (Chen et al. 2004, Afzal and Torkar 2008), but the studies on test process from the viewpoint of the organisation and the test strategy were limited. The actual research problem was elaborated from these considerations.

The research problem of this thesis is which test process components affect the test strategy and how they should be addressed in test process development. The identification of the important test process components should be done to ensure that at the organisational level all the important factors of testing are addressed. When the important factors are known, the test organisation can be developed towards better practices by removing hindrances and introducing changes, which are not in conflict with these components. Moreover, by understanding which components of the test process are important, the different test process models such as the ISO/IEC 29119 can be assessed for feasibility in a practical organisation.

As for the research approach to the organisational test strategy, the impact of different factors such as tools, methods, personnel, test case design and quality criteria required further investigation. One feasible approach was to analyse the practical impact of different components to the test process, and determine how the test strategy differs in different types of organisations. The identification of major contributing factors to the test process efficiency and perceived end-product quality would be especially helpful in allowing organisations to achieve better practices. If the identification was successful, it could also be worthwhile investigating whether there are certain test strategies for certain types of organisations which can be generalized into different template models for test strategy. Based on literature review, this approach was plausible, as the concept of "Generic Test Strategy" (De Grood 2008) already exists, and is used to define the general approach for the test process. In addition to generalization, developing the concept of preferred test strategies and identifying important test process components for test organisations was also considered beneficial outcome. In this way, the theoretical ISO/IEC 29119 model and practical testing done in organisations could have been adjoined.

The research problem, i.e. which components affect the software testing strategy and how they should be addressed in the development of test process, was decomposed into sub-problems. The specification of the concepts and additional viewpoints of the thesis (sub-problem 1) were needed to specify the scope of the thesis and give an overview of the test processes. Sub-problems 2, 3, 4 and 6 were used in the qualitative analysis of the current software test processes, concerning the emergent special questions of how different test strategy components affect the test process and the identification of the important test components from the viewpoint of the test process. In sub-problems 2 and 6, additional data was collected by applying the quantitative method to assess the sub-problem from additional viewpoints. Finally, sub-problems 5 and 7 (applying qualitative analysis) focused on assessing the test process and test process improvement as an organisational activity. The objectives of the individual studies in this thesis were derived from the specified sub-problems. Sub-problems, objectives of the studies, and the respective publications are listed in Table 3.

Table 3. Decomposition of the research problem

Sub-problem	Objective of the study	Publication
1. Which are the current problems and enhancement possibilities for software testing process?	Specification of the concepts and additional viewpoints for the thesis.	*Publication I*
2. Which methods and tools are applied on real-world software testing?	Identification and decomposition of common testing practices which are applied in real world testing.	*Publication II*
3. How organisations develop software and does the selected approach affect the testing practices?	Study the effect of development methods and agile development on the testing practices.	*Publication III*
4. How does the organisation decide on what has to be tested?	Identify the different selection methods and prioritization process of test cases in projects.	*Publication IV*
5. How and when do the organisations develop their test processes?	Analysis of the requirements and approaches applied when organisations decide to improve their existing test process.	*Publication V*
6. How do the software quality-related aspects reflect to the test process?	Analysis of the effect of quality-related aspects from the viewpoint of test process.	*Publication VI*
7. How applicable is the test standard process from the viewpoint of the real world organisations?	Development and analysis of a process improvement framework, which applies the ISO/IEC 29119 test process model.	*Publication VII*

3.2 Research subject and the selection of the research methods

This thesis applies a mixed method research combining both quantitative and qualitative research methods to approach the research subjects. The combination of different research approaches and datasets is a form of methodological pluralism (Hirschheim 1985), which itself is a way to increase the validity of the study results (Onwuegbuzie and Leech 2007). Methodological pluralism allows the researcher to observe the phenomenon from different viewpoints, and also to combine the advantages of two different approaches, such as descriptive strengths of ethnography and the analytical edge of Grounded Theory (Tan and Hall 2007). In the following, the background and reasoning for combining different research approaches is explained.

Test organisation, which by the context definitely belongs to the domain of software engineering discipline, can be considered a form of a socio-technical system (Geels 2004) as it has both social and technical aspects related to it, similarly as an information system. The socio-technical systems, as they have considerable human or social component in nature, share all the difficulties associated with the social sciences. A scientific paradigm adopted by the natural sciences is appropriate to study these types of systems only if the paradigm is also applicable in the social sciences. If one contends that the social sciences embrace a different paradigm for acquiring knowledge than their natural science counterpart, then information system study should follow the principles of the social science paradigm. (Hirschheim 1985).

Based on the Hirschheim observations, this thesis studies both social and technical aspects of the observed subjects. In a publication by Geels (2004), the social and technical aspects of software systems are considered to form a larger ecosystem, a socio-technical system (Figure 7), consisting of several elements from both social and technical disciplines which are equally important. Overall, it can be argued that the systems similar to the modern information systems consist of four different factors:

Social - norms, culture, laws, roles

Cognitive - semantics, attitudes, beliefs, ideas, morals

Information - programs, data, memory

Mechanical - hardware, computer, physical space

These should all be considered in research studying socio-technical systems. (Whitworth 2006).

In methodological pluralism, the objective is to observe the phenomenon from different viewpoints, and also to combine the advantages of two different approaches. In this study, the selected methods were a Grounded Theory study, following the principles of Strauss and Corbin (1990), and a survey (Fink and Kosecoff 1985). Methodological pluralism accepts that there is not one conclusive methodological or epistemological principle which is adequate for observing and analysing the observed phenomena. As methodological pluralism emphasises, the object of study has many different facets, and it is plausible to argue that it should be applied in studies on information systems and software engineering processes such as organisational test process.

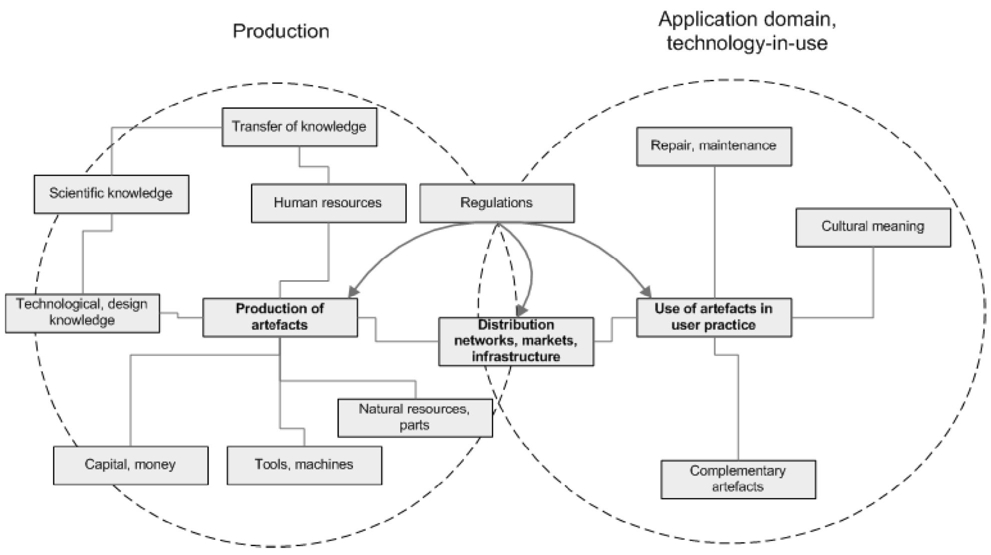


Figure 7: The basic elements of socio-technical system (Geels 2004)

The application of multiple research approaches and the triangulation of the research data also address some of the considerations as regards the different research paradigms. In the IS field, and in general studies focusing on large ecosystems, this consideration is the epistemological assumptions, namely that between the positivist and interpretivist paradigms i.e. the quantitative and qualitative approaches. In positivist research, the research results, in theory, claim intersubjective validity between the results, whereas in the interpretivist approach the focus is on selecting and applying appropriate research methods to understand the study subjects individually (Becker and Niehaves 2007). Traditionally, positivism is usually associated with the quantitative approaches, while interpretivism is closely associated with qualitative methods (Chen and Hirschheim 2004). In some contexts, the epistemological differences may even cause the research, which observes the same phenomenon, not to be mutually understood, as some theories and constructs seem different between these different viewpoints, such as theory on activity and the social implementation of that said activity, or simply the way the researcher observed the phenomena (Weber 2003).

To address this issue, a term of critical realism has been defined (For example Carlsson 2003, Pather and Remenyi 2004). Critical realism research aims to utilise both qualitative and quantitative methods and strive beyond such limitations to explain and understand the subjects of the study (Pather and Remenyi 2004). Critical realism draws on the existing approaches, not committing to a single form of research, but aiming to understand the object from all different aspects, while acknowledging the possible fallibilities and limitations of their research (Mingers 2004, Pather and Remenyi 2004). Therefore, the combination of both qualitative and quantitative research is applied in this study to understand the research subjects and their implementations of the test process components in detail, while still retaining the ability to conduct intersubjective analysis between the research subjects on the test process concepts and thesis topics.

3.2.1 The research subject

In this thesis, the ISO/IEC 12207 Software life cycle processes (ISO/IEC 2008) standard was initially used to describe the research subjects, software-producing organisations and their product outputs. In ISO/IEC 12207 the organisation and products are described to compose a set of processes. This definition was expanded in the testing-related processes with the test process model defined in the ISO/IEC 29119 Test Standard (2010), which defines the organisation and process activities from the viewpoint of testing. In addition of organisation model, ISO/IEC 15504-1 (ISO/IEC 2002) was applied to define the fundamental concepts for process improvement, as this standard offers an assessment model for organisational processes defined in the ISO/IEC 12207. These standards formed the a priori understanding of the research subjects, defining the different activities and components the subject organisation was considered to have.

In addition to the process models for development and testing, the definition of software quality was taken from the standard ISO/IEC 25010 Software product quality requirements and evaluation quality model (ISO/IEC 2009) to enable the study to assess the output of the research subjects. In this model, the software quality is defined as an amalgam of eight quality characteristics, which each have a number of objectively measurable or evaluable sub-characteristics, which describe the software specific activities and the system context on which the software is developed and maintained.

From ISO/IEC 15504, a concept of an organisational unit (OU) was also derived to define the organisational subsets studied in this research. As defined in ISO/IEC 15504-1 Concepts and Vocabulary (2002), an organisation unit is a part of an organisation, which deploys one or more processes with coherent processes context and operates within a coherent set of business goals. An organisational unit can consist of one specific project or a specific group responsible for one product within a larger corporation, but especially in micro and small-sized (EC 2003) companies, one organisation unit can consist of the entire company. In larger organisations, an OU operates mostly independently, but receives some amounts of organisational level steering from the upper management. In smaller organisations, the organisational management operates within the OU or is directly above it. As the large companies may have different business goals than the small companies, it was unfeasible to compare them directly; similarly different projects may have different purposes and goals. The reason to apply OUs as an assessment unit instead of entire corporations or projects was to normalise the differences between the organisations, and minimise the effect of different objectives and business goals, and to enable direct comparison between research subjects.

3.2.2 The selection of the research methods

In Grounded Theory (Glaser and Strauss 1967), the objective of the research is to present an accurate description of what is being studied, and by methods of reduction and generalisations to build a believable descriptive narrative and chain of evidence from observations to a descriptive model with little or no interpretation on the studied phenomenon (Strauss and Corbin 1990). The Grounded Theory method allows the research question freedom to explore the phenomenon in depth, and allows a broader viewpoint on the topic than quantitative approaches. The Grounded Theory method was selected as an analysis method in the preliminary phase of the study, as the nature of the research topic, and the existing data, was considered too broad and unstructured for quantitative analysis. This method was considered appropriate, as the Grounded Theory method is in general considered suitable to uncover and understand complex phenomena founded on large ecosystems and gain novel and fresh viewpoints on areas, which are otherwise generally well-known (Strauss and Corbin 1990).

In the main data collection and analysis phase, the Grounded Theory method was applied as it suited the purposes of the study as the research topic, test processes in organisations, was considered a large and broad topic. The concept of conducting the study by using some form of action research (for example, Susman and Evered 1978) was rejected as the possibility of affecting the organisations and studying the effect of the changes, which forms the core of the action research approach, was limited.

On selection of the Grounded Theory, the second decision was then between the disciplines of Glaserian (outlined in Glaser 2002, van Niekerk and Roode 2009) and Strauss-Corbin (1990) approaches. The Strauss-Corbin-approach focuses on coding paradigms and in systematic categorisation and analysis of the collected data to uncover the relevant factors behind observed phenomena, whereas the Glaserian approach focuses on passive observation and emergence of strong codes from the data which then can be used to identify the relevant factors. In the preliminary phase, Strauss-Corbin was applied on the analysis of the existing data because of its codification method, which allowed detailed and structured analysis on the collected qualitative data set. In the latter phases, the Strauss-Corbin-method was applied as the number of organisations participating in the study was relatively high for a qualitative study, and the possibilities of passively and comprehensively observing the twelve organisations to the degree required by the Glaserian approach was considered unfeasible. Although the Glaserian approach is also a merited and appropriate method, the practical limitations and applicability in our research context made the Strauss-Corbin more suitable for the study purposes, and therefore it was applied throughout the research process.

In addition to the qualitative study using the Grounded Theory approach, quantitative data was collected from a survey (Fink and Kosecoff 1985). The survey method is an appropriate method to collect data from a standardised group of personel, such as software development professionals such as software project leaders and test managers. According to Pfleeger and Kitchenham (2001), a survey is a comprehensive method for collecting specific and measurable information to describe, compare or explain knowledge, attitudes and behaviour.

The survey was also selected as an additional research method for the study to enable triangulation of research data (Denzin 1978). The triangulation of data in research means application and comparison of several types and sources of data to further validate the results. According to the literature (Seaman 1999, Paré and Elam 1997), the combination of quantitative and qualitative methods is usually more beneficial than applying either approach separately: statistical relationships found between the quantitative variables can be verified against qualitative data and vice versa. In this study, the qualitative data collected with the interviews and quantitative data collected with survey enabled the comparison between the data sources and was applied to further validate the results, as demonstrated in *Publication II* and *Publication VI*.

3.3 Research process

The research process was divided into three phases. In the preliminary phase of the thesis, the Grounded Theory method was applied on the previously collected interview data, along with a literature review on the relevant topics to establish basic understanding of the research area. Additional research topics were collected from the expert group on the software testing, consisting of software engineering researchers and industry representatives. In the second phase, the main data collection and analysis, the research methods were a qualitative analysis using the Grounded Theory method on collected interview data, supplemented with a quantitative survey. In the third phase, the validation phase, the observations from the earlier phases were studied with additional interviews and subsequent Grounded Theory analysis. The research process, along with the different research phases, is illustrated in Figure 8.

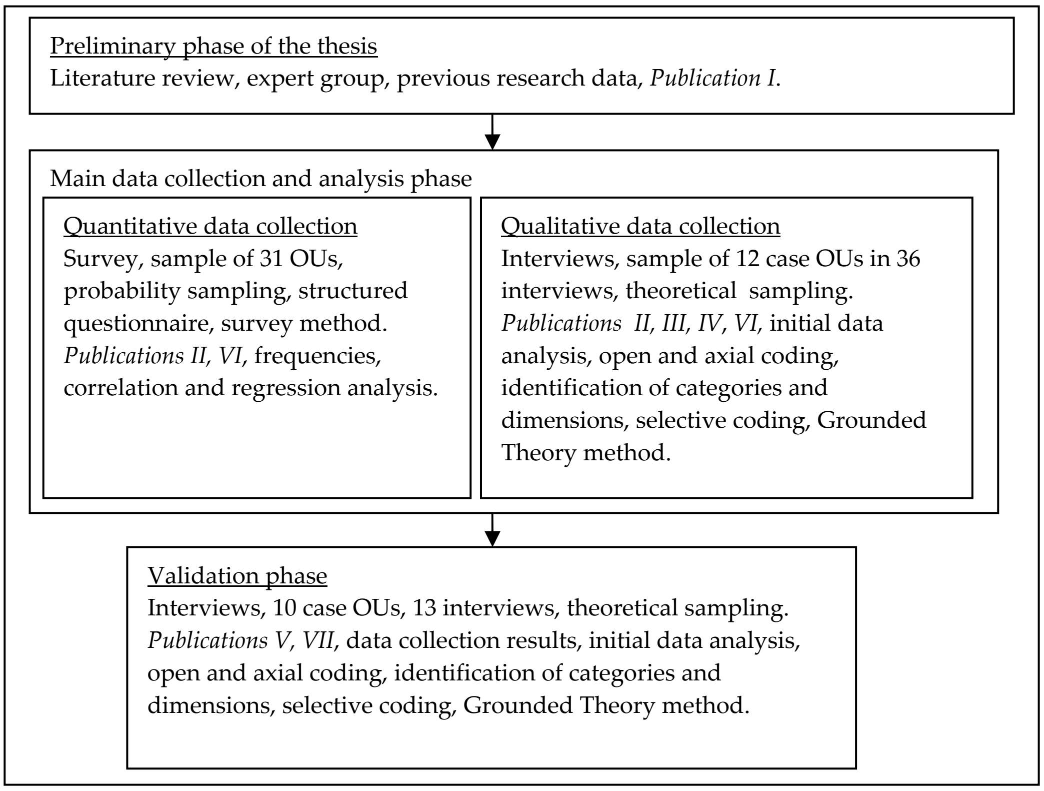

Preliminary phase of the thesis
Literature review, expert group, previous research data, *Publication I.*
Main data collection and analysis phase
Quantitative data collection
Survey, sample of 31 OUs, probability sampling, structured questionnaire, survey method. *Publications II, VI,* frequencies, correlation and regression analysis.
Qualitative data collection
Interviews, sample of 12 case OUs in 36 interviews, theoretical sampling. *Publications II, III, IV, VI,* initial data analysis, open and axial coding, identification of categories and dimensions, selective coding, Grounded Theory method.
Validation phase
Interviews, 10 case OUs, 13 interviews, theoretical sampling. *Publications V, VII,* data collection results, initial data analysis, open and axial coding, identification of categories and dimensions, selective coding, Grounded Theory method.

Figure 8. Research process and phases

3.3.1 Preliminary phase of the thesis

During the preliminary phase of the thesis, an extensive literature review was done to better understand the test processes and search for categories of interest and specify the viewpoints of the thesis. In addition to the literature review, the existing data from previous research project ANTI, reported in (Karhu et al. 2009, Taipale and Smolander 2006, Taipale et al. 2006a, 2006b, 2006c), were examined to establish basic understanding over real-life-testing and find appropriate seed categories (Miles and Huberman 1994). The ANTI project was a software engineering research project conducted by some of the researchers from our laboratory. The ANTI project focused on test process costs and quality factors, applying first the Delphi method (Schmidt 1997) and then collecting more detailed data with the Grounded Theory approach. More details on the ANTI research process and the results of the preliminary phase of the thesis are reported in *Publication I.*

The previous research data was collected from five organisational units (OUs, see Table 2) which participated in the previous research project ANTI. This data which consisted of interview recordings, transcriptions, earlier codifications and interview memos, was codified according to the Strauss & Corbin Grounded Theory principles to identify strong categories in the interview themes of test process problems and enhancement proposals.

Table 2. Analysed organisations from the preliminary phase

Business	Company size	Interviewed personnel
A MES producer and integrator	Large/international	Testing manager, tester, systems analyst
Software producer and testing service provider	Small/national	Testing manager, tester, systems analyst
A process automation and information management provider	Large/international	Testing manager, tester, systems analyst
Electronics manufacturer	Large/international	Testing manager, 2 testers, systems analyst
Testing service provider	Small/national	Testing manager, tester, systems analyst

3.3.2 Main data collection and analysis phase of the thesis

In the main data collection and analysis phase, the focus of the research was on collecting data on a large, heterogeneous group of real-life software organisations to understand how software testing in real life works. The areas of interest were to test whether the a priori constructs such as literature review and *Publication I* results were still valid, and in collecting data on testing-related aspects in both software development and in the testing itself. The data collection was done with two main approaches intended to complement each other. Qualitative data was collected for the Grounded Theory analysis in twelve "focus group" organisations based on theoretical sampling, and quantitative data was collected with a survey from 31 organisations, which were selected on supplementing the "focus group" with probability sampling.

Data collection

The beginning of a qualitative study includes the definition of a research problem, possible a priori constructs, the selection of cases, and the crafting of instruments and protocols for data collection (Eisenhardt 1989). The prior literature and research data, in which 30 different software companies were interviewed and 5 subsequently analysed in detail, were used in the initial design of research. The definition of priori constructs and the selection of polar points was also based on the earlier ANTI research project results and experiences, in terms of selection of the representative cases.

Furthermore, the case selection criteria was set to include only organisation units, which as their main type of business activity develop software or provide software process-related services in a professional manner. Furthermore, on order to limit a possible company bias, the number of participating organisation units was limited to one OU per company, even if some larger companies could have participated with several different OUs. According to Eisenhardt (1989), this approach is feasible. In addition, in inductive theory building the a priori data should not affect the tested theories or hypotheses. Therefore, no particular emphasis was put on the pre-existing data or formal standard definitions when observing and analysing the studied organisations.

For the case study, twelve OUs were selected as the "focus group" (see Table 3) based on the previous results and identified domain types. The sampling was theoretical (Paré and Elam 1997) and the cases were chosen to provide examples of polar types (Eisenhardt 1989), which meant that the cases represented different types of OUs, with differences in the business area, size of the company and market size. Theoretical sampling (Glaser and Strauss 1967) describes the process of choosing research cases to compare with other cases. The goal of theoretical sampling is not the same as with probabilistic sampling; the goal is not to collect representative sample of the entire population, but to gain a deeper understanding of the analysed cases and identify concepts and their relationships. In practice, the organisations were selected from a group of research partners and collaborators, and supplemented with additional organisations to represent organisation types not present. The actual data collection instruments were theme-based questionnaires and a survey, available as Appendixes II and III.

The data collection phase included three theme-based interview rounds, of which the second combined both qualitative and quantitative aspects. The companies were visited personally and 36 recorded interviews were carried out for the case OUs of the qualitative research, and an additional 19 interviews for the quantitative analysis to achieve the requirements of statistical relevance. The duration of the interviews varied between one and one and a half hours and they were all tape-recorded and transcribed. The interviews were conducted under partial confidentiality agreement by the project researchers to ensure that the interviewees understood the questions correctly and could openly discuss matters that could potentially jeopardize trade secrets. Under this partial confidentiality agreement, no full source data would be publicly available, but partial, anonymized compositions of the interview data could be used in the publications. A memo containing the issues emphasised was also written during the interviews.

Table 3. Analysed organisations from the main data collection and analysis phase

OU	Business, typical product type	Company size / Operation	Amount of agile practices[1]
Case A	MES producer and electronics manufacturer, embedded software for hardware product	Small / National	Low
Case B	Logistics software developer, software for hardware system	Large / National	High
Case C	ICT consultant, service producer	Small / National	Low
Case D	Internet service developer and consultant, service producer	Small / National	Low
Case E	Maritime software system developer, software product	Medium / International	Medium
Case F	Safety and logistics system developer, software for hardware system	Medium / National	Low to none
Case G	Financial software developer, software product	Large / National	Low to none
Case H	ICT developer and consultant, embedded software for hardware product	Large / International	Low to none
Case I	Financial software developer, software product	Large / International	Low
Case J	SME business and agriculture ICT service provider, software product	Small / National	Medium
Case K	MES producer and logistics service systems provider, embedded software for hardware product	Medium / International	Medium
Case L	Modeling software developer, software product	Large / International	Low
19 survey-only cases	Varies; from software consultancies to software product developers and hardware manufacturers.	Varies	Varies

[1]See *Publication III* for more details

The first interview round that was completed during the qualitative analysis served also as the review for the quantitative interview themes. The first interview round contained only semi-structured (open) questions, and the objective was to understand the basic practice of testing, identify the central themes for the next round, and in general, identify central concepts and factors of the test process in the real-life organisations. The interviewees were software or architecture developers or test designers. In some interviews, there was more than one interviewee present, for example a software developer and architecture developer. Such interviews usually lasted more than one hour. The questions on the first round were themed around the basics of the OU testing process, testing resources, software development processes and testing environment.

The interviewees in the second round were test managers or project leaders responsible for software projects. As earlier, the duration of the interviews varied between one and one and half hours and consisted of a survey and a supplemental set of semi-structured interviews, conducted by researchers working on the project. The objective of the second interview round was to achieve deeper understanding of the software testing practice and gain formal information on company testing framework and practices. The interviewees were selected to be managers and leaders because it was considered that they were more capable of assessing the test process from the viewpoint of the entire organisation.

The questions were theme-based and concerned problems in testing, the utilisation of software components, the influence of the business orientation, communication and interaction, schedules, organisation and know-how, product quality aspects, testing automation, and economy. The structure of the questions varied from structured survey questions to supplemental, semi-structured, open questions. From the 19 interviews with the organisations only participating in the survey, the semi-structured interview answers were not included in the qualitative data analysis as they lacked the context and additional information regarding the organisation collected from other interview rounds.

In the third interview round the interviewees were testers or programmers who had extensive testing responsibilities in the same OUs that were interviewed during the first and second round. Once again, in the third round, the interviews were held by the researchers to ensure that the interviewees understood the questions correctly and that all of the questions were answered to a satisfactory degree. The interviews in this round focused on topics such as problems in testing – complexity of the systems, verification, testability – the use of software components, testing resources, outsourcing and customer influence in the test process. A full list of interview themes and a description of the interviewee roles are listed in Table 4.

Table 4. Data collection rounds in the main data collection and analysis phase

Round type	Number of interviews	Interviewee role	Description	Themes
1) Semi-structured interview	12 focus OU interviews	Designer or Programmer	The interviewee was responsible for or had influence on software design.	Design and development methods, Testing strategy and methods, Agile methods, Standards, Outsourcing, Perceived quality
2) Structured survey with Semi-structured interview	31 OUs, including 12 focus OUs	Project manager or Testing manager	The interviewee was responsible for the sofware project or testing phase of the software product.	Test processes and tools, Customer participation, Quality and Customer, Software Quality, Testing methods and -resources
3) Semi-structured interview	12 focus OU interviews	Tester or Programmer	The interviewee was a dedicated tester or was responsible for testing the software product.	Testing methods, Testing strategy and resources, Agile methods, Standards, Outsourcing, Test automation and services, Test tools, Perceived quality, Customer in testing

In two of the first round interviews, the organisation elected two people for the interview, as they considered that they do not have any individual worker, whose responsibilities match with the desired interviewee role. Additionally, on one occasion, the organisation was allowed to supplement their earlier answers in a later interview as the interviewee thought that the original answers lacked some crucial details.

Data analysis with the Grounded Theory

The grounded analysis was used to provide insight into the software organisations, their software processes and testing activities. By interviewing people in different positions from the software organisation, the analysis could gain additional information on testing-related concepts, such as different testing phases, test strategies, testing tools and case selection methods. Later this information was compared between organisations, allowing hypotheses on the test process components from several viewpoints and from the test process itself as a whole.

The Grounded Theory method contains three data analysis steps: open coding, axial coding and selective coding. The objective for open coding is to extract the categories from the data, whereas axial coding identifies the connections between the categories. In the third phase, selective coding, the core category is identified and described (Strauss and Corbin 1990). In practice, these steps overlap and merge because the theory development process proceeds iteratively. Additionally, Strauss and Corbin state that sometimes the core category is one of the existing categories, and at other times no single category is broad enough to cover the central phenomenon. In *Publications I, II, III* and *VI* the core category of the observed test concept was identified to be such an umbrella category, whereas in *Publications IV, V* and *VII* the core category was identified to be an existing or specific category from the research data.

The objective of open coding is to classify the data into categories and identify leads in the data, as shown in the Table 5. The interview data was classified into categories based on the main issue, with any observation or phenomenon related to it being the codified part. In general, the process of grouping concepts that seem to pertain to the same phenomena is called categorising, and it is done to reduce the number of units to work with. In this study, this was done using ATLAS.ti software (ATLAS.ti 2011) which specialises on the analysis of qualitative data. The open coding process started with "seed categories" (Miles and Huberman 1994) that were formed from the research sub-question the publication was studying and prior observations from the earlier publications. Overall, the analysis process followed the approach introduced by Seaman (1999), which notes that the initial set of codes (seed categories) come from the goals of the study, the research questions, and predefined variables of interest. In the open coding, we added new categories and merged existing categories into others, if they seemed unfeasible or if we found a better generalisation.

After collecting the individual observations into categories and codes, the categorised codes were linked together based on the relationships observed in the interviews. For example, the codes "Software process: Acquiring 3rd party modules", "Testing strategy: Testing 3rd party modules", and "Problem: Knowledge management with 3rd party modules" were clearly related and therefore could be connected together in the axial coding. The objective of axial coding is to further develop categories, their properties and dimensions, and find causal, or any other kinds of connections between the categories and codes. For some categories, the axial coding can also include actual dimension for the phenomenon, for example "Personification-Codification" for "Knowledge management strategy", or "Amount of Designed Test Cases vs. Applied" with dimension of 0-100%, where every property could be defined as a point along the continuum defined by the two polar opposites or numeric values. Obviously, for some categories, which were used to summarise different observations such as enhancement proposals, opinions on certain topics or process problems, defining dimensions was unfeasible. At first using dimensions for some categories was considered, for example, "criticality of test automation in testing process" or "tool sophistication level for automation tools" in the analysis, but later they were discarded as they were considered superficial, obfuscated the actual observation, and overall, yielded only little value in cases where the dimension was not apparent.

Table 5: Example of codification process

Interview transcript	Codes, Category: Code
"Well, I would hope for ***stricter control or management for implementing our testing strategy****, as I am not sure if our* ***testing covers everything and is it sophisticated enough****. On the other hand, we do have* ***strictly limited resources, so it can be enhanced only to some degree****, we cannot test everything. And perhaps, recently we have had, in the newest versions, some regression testing, going through all features, seeing if nothing is broken, but* ***in several occasions this has been left unfinished because time has run out****. So there, on that issue we should focus."*	Enhancement proposal: Developing testing strategy Strategy for testing: Ensuring case coverage Problem: Lack of resources Problem: Lack of time

Our approach to analysis of the categories included Within-Case Analysis and Cross-Case-Analysis, as specified by Eisenhardt (1989). Basically, this is a tactic of selecting dimensions and properties with within-group similarities coupled with inter-group differences based on the comparisons between different research subjects. In this strategy, one phenomenon that clearly divided the organisations into different groups was isolated, and looked into for more details explaining differences and similarities within these groups. As for one central result, the appropriateness of OU as a comparison unit was confirmed based on our size difference-related observations on the data; the within-group- and inter-group comparisons did yield results in which the company size or company policies did not have strong influence, whereas the local, within-unit policies did. In addition, the internal activities observed in OUs were similar regardless of the originating company size, meaning that in this study the OU comparison was indeed a feasible approach.

Each chain of evidence was established and confirmed in this interpretation method by discovering sufficient citations or finding conceptually similar OU activities from the case transcriptions. Finally, in the last phase of the analysis, in selective coding, the objective was to identify the core category – a central phenomenon – and systematically relate it to other categories and generate the hypothesis and the theory. Overall, in theory building the process followed the case study research described by Eisenhardt (1989) and its implementation examples (Klein and Myers 1999, Paré and Elam 1997).

The general rule in Grounded Theory is to sample until theoretical saturation is reached. This means, until (1) no new or relevant data seem to emerge regarding a category, (2) the category development is dense, insofar as all of the paradigm elements are accounted for, along with variation and process, and (3) the relationships between categories are well established and validated (Strauss and Corbin 1990). In this study, saturation was reached during the third round, where no new categories were created, merged or removed from the coding. Similarly, the attribute values were also stable, i.e. the already discovered phenomena began to repeat themselves in the collected data. As an additional way of ensuring the validity of the study and in order to avoid validity threats, four researchers took part in the data analysis. The bias caused by researchers was reduced by combining the different views of the researchers (observer triangulation) and a comparison with the phenomena observed in the quantitative data (methodological triangulation) (Denzin 1978).

Data analysis with the survey instrument

In the quantitative parts of the study, the survey method described by Fink and Kosecoff (1985) was used as the research method. According to Fink (2003), a sample is a portion or subset of a larger group called a population, which includes all organisations which are potential survey respondents. The sample in the survey should aim to be a miniature version of the population, having the same consistency and representatives for all relevant domain types, only smaller in size. In this study, the population consisted of organisational units as defined in ISO/IEC 15504-1. The sample was constructed by taking the focus group collected for the qualitative analysis, and supplementing it with probability sampling (see Fink & Kosecoff 1985) to have sufficient statistical relevance, following principles presented by Iivari (1996). In practice, the probability sampling was done by expanding the sample with 19 additional organisations, collected from the university and research group company contacts by random selection and confirming by a phone call that the organisation fitted the sample criteria. Out of a total of 30 organisations that were contacted, 11 were rejected based on this contact, as they either did not fit the sample criteria or decided not to participate on the study.

For the selected approach, the actual methods of data analysis were partially derived from Iivari (1996). He surveyed computer-aided software engineering tool adoption. The sample was 109 persons from 35 organisations. He derived the constructs from the innovation diffusion/adoption theory. Iivari estimated the reliabilities of the constructs using Cronbach coefficient alpha (Cronbach 1951). In factor analysis, he used principal component analysis (PCA) and in data analysis regression analysis. We used also used Cronbach alpha for measuring the reliabilities of the constructs consisting of multiple items and in comparisons of the correlations between different constructs with Kendall's tau_b correlation. In these calculations, a specialised statistical analysis software SPSS (SPSS 2011) was used.

A validated instrument increases the reliability of the measurements, but such an instrument was not available in the literature, so we designed our own interview instrument based on the questionnaire derived from Dybå (2004). This questionnaire was an instrument for measuring the key factors of success in software process improvement, which we in our study adapted to study the perceived end-product quality and the effect of different quality-related factors in software testing.

Related surveys can be categorised into two types: Kitchenham et al. (2002) divide comparable survey studies into exploratory studies, from which only weak conclusions can be drawn, and confirmatory studies, from which strong conclusions can be drawn. This survey belongs to the category of exploratory, observational, and cross-sectional studies as our intention was to study the different identified factors and observe their effect on the test process and end-product quality.

The survey was conducted at the second interview round during the face-to-face interviews. A few open-ended questions were located at the end of the questionnaire to collect data for the qualitative study. The questionnaire was planned to be answered during the interview to avoid missing answers because they make the data analysis complicated, for example, for the calculation of correlation. For these reasons, a self-assisted, mailed questionnaire was rejected and personal interviews were selected. In addition, as Baruch (1999) has stated, the response rate for academic surveys is usually less than two thirds depending on the surveyed population. Had the survey been a mail-in questionnaire instead of an interview, it would have been probable that besides missed questions, the sample size would have been even smaller. The questionnaire was also piloted with three organisations and four private individuals before the actual data collection round to test the form and the questions for clarity and understandability.

3.3.3 Validation phase of the study

In the validation phase of the thesis study, the focus shifted from the identification of testing work-effecting process components to the entire process organisation. In this phase, the test process of the organisation, and subsequently the concepts of test process improvement were studied. The objective was to understand how the identified test process components should be addressed at an organisational level. Additional concern was to test the feasibility of the ISO/IEC 29119 test process model and develop a framework for organisations to develop their test process towards better practices and conformance with the principles presented at the standard-defined test process model.

Data collection

The validation phase of the study had a new set of data collection interviews with a partially new group of participating organisations. Otherwise the interviews were organised similarly, as in the main data collection and analysis phase interview rounds one and three. The fourth round interviewees were test managers, as their viewpoint was considered, from the project-level organisation, the most suitable to assess and discuss the observations from earlier rounds and to assess the applicability of the standard process model within the organisations. The interviews were theme-based, including questions from themes such as test strategy, test policy, test planning, testing work in general, software architecture, and crowd sourcing. These interviews were held in cooperation with another dissertation study, so some of the interview themes were not directly directed at this dissertation work. A list of interviewed organisations is available as Table 6.

Table 6. Analysed organisations from the validation phase

OU	Business domain, product type	Company size / Operation domain
Case A*	ICT developer and consultant, service producer	Small / National
Case B*	Safety and logistics systems developer, software products	Medium / National
Case C	Financial and logistics software developer, software products	Medium/ National
Case D*	MES producer and logistics system provider, embedded software for hardware products	Medium / International
Case E*	MES producer and electronics manufacturer, embedded software for hardware products	Small / National
Case F*	Maritime software systems developer, software products	Medium / International
Case G	ICT consultant specialicing in testing, test consulting services	Medium / National
Case H*	Modeling software developer, software products	Large / International
Case I*	ICT developer and consultant, software production consulting	Large / International
Case J	ICT consultant specialicing in testing, test consulting services	Small / National

* This organisation also participated in interview rounds 1-3

In addition to the fourth round of interviews, a validation step of *Publication VII* also included a study on four organisations based on the prior interview data. To confirm the findings of this study, three of the organisations were interviewed to review and collect feedback on the study results. A fourth organisation was offered the opportunity, but due to the changes in their organisation, they declined to participate in this part. Additionally, one interviewee from the fourth round interviews cancelled the interview for personal reasons, but provided written answers by email.

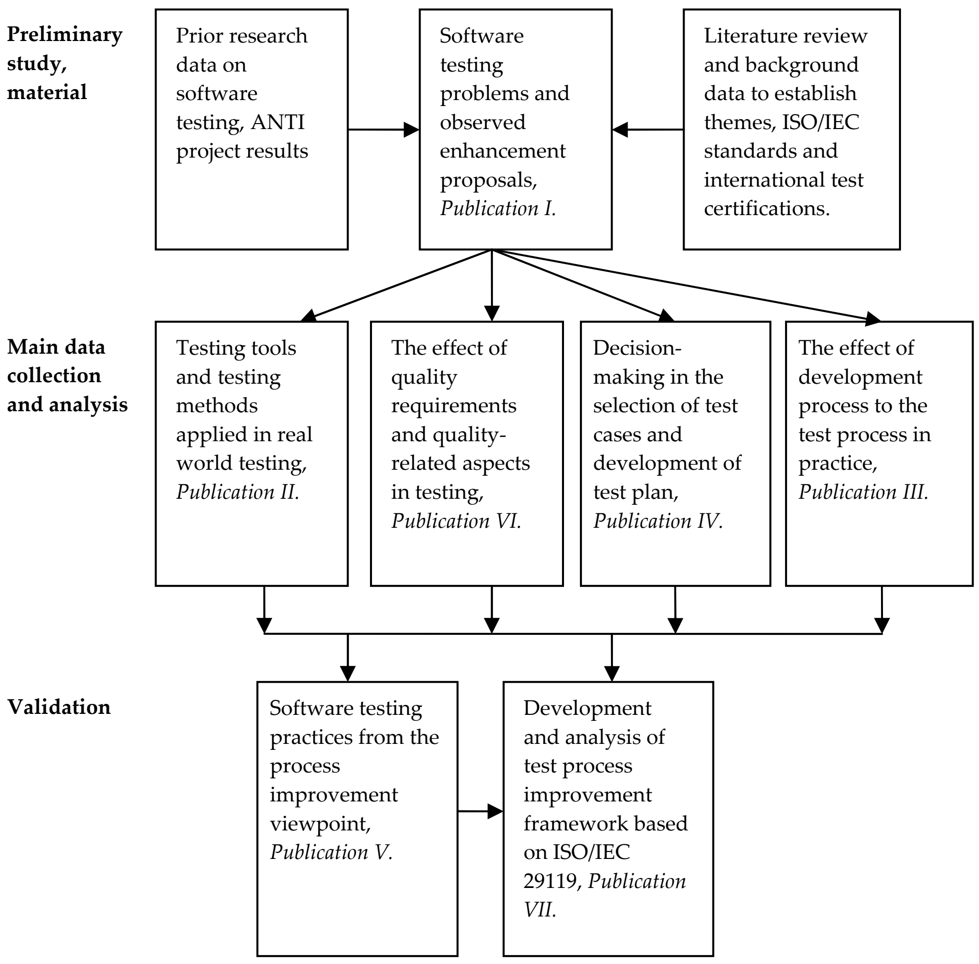
Both of the interview sets, 4th interview round and validation interviews for *Publication VII*, were analysed with the Strauss-Corbin Grounded Theory -approach, similarly to the previous research phase.

3.3.4 Finishing and reporting the thesis

Each phase of the thesis answered its part of the research problems, but also raised new questions, which were addressed in the upcoming phases. In the preliminary phase, the scope of the thesis was on identifying the potentially interesting and important test process components from the previously collected data (*Publication I*), literature review and software standards.

In the main data collection and analysis phase, the organisations participating in the study were interviewed in three consecutive rounds to collect data on themes related to the test process. The themes were selected based on previous study phase results, literature review and topics of interest identified from the interviews themselves. During the second interview round, a quantitative survey was also conducted in all of the participating organisations, and in an additional 19 organisations participating solely in the survey. The data collected during this study phase was published in several publications. The overall testing resources and testing approach were discussed in *Publication II*, which combined both the survey and interview results. In *Publication III*, the effect of different development methods and the overall influence of the agile principles in testing were assessed, while *Publication IV* focused on the aspects related to the project-level management, test case selection and development of a test plan. *Publication VI* combined both qualitative and quantitative data to study the quality concepts and perceived quality in the software organisations and the effect different testing activities may have on the perceived end-product quality.

The last phase of the study was the validation phase, where the organisations were studied as a whole in order to study the feasibility of the ISO/IEC 29119 test process model and establish a framework to allow organisations to identify possible development needs and develop their test process activities towards better practices. These topics are discussed in two publications. *Publication V* studies how the organisations develop their test practice or adopt new testing methods as well as how applicable the proposed standard model is in a real-life organisation. The final publication of this dissertation, *Publication VII*, introduces a proof-of-concept for a self-assessment framework which was developed based on the study observations. This framework allows organisations to assess their current testing practices and develop their test process towards concepts and activities presented in the ISO/IEC 29119 standard. The progress of this thesis work and the relationships between the different studies included in this dissertation are illustrated in Figure 9.



Preliminary study, material
Prior research data on software testing, ANTI project results
Software testing problems and observed enhancement proposals, *Publication I.*
Literature review and background data to establish themes, ISO/IEC standards and international test certifications.
Main data collection and analysis
Testing tools and testing methods applied in real world testing, *Publication II.*
The effect of quality requirements and quality-related aspects in testing, *Publication VI.*
Decision-making in the selection of test cases and development of test plan, *Publication IV.*
The effect of development process to the test process in practice, *Publication III.*
Validation
Software testing practices from the process improvement viewpoint, *Publication V.*
Development and analysis of test process improvement framework based on ISO/IEC 29119, *Publication VII.*

Figure 9. Products and relationships between the thesis publications

3.4 Summary

The research phases and their essential methodical details and constructs are summarised below in Table 7.

Table 7. The research phases

Phase	**Preliminary phase**	**Main data collection and analysis**	**Validation phase**
Research problem	What are the most pressing test process problems in real-life organisations? Viewpoints for the thesis.	Which test process factors have the most influence on the test work itself? Which test process activities affect the perceived end-product quality? Affecting factors and their relationships.	How do organisations develop their test processes? Is the standard test process model feasible in real-life organisations? Development of a framework to assess test process maturity and identify development objectives.
A priori constructs	ANTI-project results.	Viewpoints of the thesis, ISO/IEC 29119 process model.	Viewpoints of the thesis, affecting factors and their relationships, ISO/IEC 29119 process model.
Case selection/ interviewees	Steering group, expert group.	12 OUs in the qualitative analysis, 31 OUs in the quantitative survey.	10 OUs in process improvement study, 4 OUs in development of self-assessment framework.
Instruments and protocols for data collection	Literature review, interviews.	Interviews, semi-structured questions, survey, structured questionnaire.	Interviews, semi-structured questions.
Data analysis	Qualitative analysis with ATLAS.ti software	Statistical analysis with SPSS software, qualitative analysis with ATLAS.ti software .	Qualitative analysis with ATLAS.ti software.
Applied data set	20 interviews with 4 OUs from ANTI-project.	36 qualitative interviews with 12 focus OUs and a survey of 31 OUs	13 supplemental interviews, 36 qualitative interviews with 12 focus OUs and a survey of 31 OUs.
Reporting	*Publication I*	*Publications II-IV, VI*	*Publications V and VII*

4 Overview of the publications

In this chapter an overview, and the most important results, of the included thesis publications are shortly introduced. Besides this chapter, the results of this research are presented in detail in the appendix consisting of the seven publications, in original publication form and in full length.

The publications included to this thesis have been published separately in scientific venues, which have all employed a peer-review process before acceptance for publication. In this chapter, each of these publications, their objectives, results, and relation to the whole, are discussed. The contents of these publications can be condensed with the following objectives of the studies:

- *Publication I*: Overview of the real-life concerns and difficulties associated with the software test process.
- *Publication II*: Overview of the testing resources and testing methods applied to real-life test organisations.
- *Publication III*: Analysis of the effects the applied development method has on the test process.
- *Publication IV*: Analysis of the test case selection and test plan definition in test organisations.
- *Publication V*: Analysis of the requirements for developing test process or adopting new testing methods in software organisations.

- *Publication VI*: Analysis of associations between perceived software quality concepts and test process activities.
- *Publication VII*: Introduction of a test process assessment framework combining maturity levels and ISO/IEC 29119 standard test process model.

In the following, the publications are summarised based on the objectives, results and impact as regards the whole thesis study.

4.1 Publication I: Overview of the real-life concerns and difficulties associated with the software test process

4.1.1 Research objectives

The objective of this Grounded Theory study (Strauss & Corbin 1990, Glaser & Strauss 1967) was to reveal important testing process issues and generate insights into how the testing processes could be enhanced from the viewpoint of the organisations, and what factors in testing seem to be the most usual problematic areas.

4.1.2 Results

The results indicate that the main components associated with testing process difficulties are most likely caused by the testing tools, knowledge transfer, product design, test planning, or test resource issues. According to the results, standardisation and automation levels in test process are not very high, and all cases the OUs had several enhancement proposals for immediate improvements in test processes. Similarly, it reinforced assumption that OU level comparisons between different sizes and types of organisations are feasible, as the results indicated similar issues regardless of the company of origin. Based on these results our study was able to pinpoint several key issues that were incorporated into the categories of interest in the following phase, and also gave insight on the testing infrastructure and operational framework of a real-life test organisation.

4.1.3 Relation to the whole

The results of this preliminary study was to examine the existing data on software organisations, to identify the test process components, and collect possible lead-in seed categories (Miles and Huberman 1994) for the main data collection and validation phase. Additionally, this preliminary publication was used to assess the feasibility of applying the Grounded Theory approach to the data analysis, even though the existing theory (Strauss and Corbin 1990) along with the studies by Sjoberg et al. (2007) and Briand and Lapiche (2004) supported the empirical observations on the test process research.

The results indicated several possible weaknesses in the test processes, such as, resource availability and allocation, weak testability of the software product, and testing tool limitations. The results also identified several possible enhancement proposals in addition of process hindrances, although interestingly the enhancement proposals and difficulties did not always intersect with each other. The study also confirmed that in qualitative studies, different types of organisations could be studied and compared against each other by conducting the study on the organisation units (OU). Additionally, the study results indicated that an organisational study on software test process could be fruitful; most of the identified issues could have been handled by designing a better organisational approach, for example, by introducing test and resourcing plans. Overall, the generated hypotheses and results of the literature review in this publication were applied later in the development of the data collection questionnaires.

4.2 Publication II: Overview of the testing resources and testing methods applied in real-life test organisations

4.2.1 Research objectives

The objective of this mixed method study combining both the Grounded Theory method (Strauss and Corbin 1990, Glaser and Strauss 1967) and statistical analysis was to examine and identify the current state of testing tools and test automation in the software industry. Another objective was to examine what types of software testing are performed in the professional software projects, and what percentage of total development resources are dedicated to software testing.

4.2.2 Results

The results presented further evidence on the practical test work, indicating that the test processes in organisations are defined but in many cases, not in a very formal way. Based on the results, it was established that majority of the organisations did have an established procedures which could be understood as a formal test process but in several cases these processes were only generally agreed principles or otherwise very open to interpretation. The organisations on average dedicated one fourth of their resources to the testing tasks, although variance between individual organisations was considerable. In a few organisations the test process was considered to be fully resourced, whereas other organisations reported that as low as 10 percent of the optimal resource needs were available. The test resource results are indicated in Table 8.

Table 8. Testing resources available in software organisations

	Max.	Min.	Median
Percentage of automation in testing.	90	0	10
Percentage of agile (reactive, iterative) vs. plan driven methods in projects.	100	0	30
Percentage of existing testers vs. resources need.	100	10	75
Percent of the development effort spent on testing	70	0	25

As for the test tools and test automation, it was evident that automation is a costly investment, which can be done correctly but requires dedication and continuous commitment from the organisation in order to succeed. It was also established that most of the organisations do have testing-dedicated tools, the most common groups being test management tools, unit testing tools, test automation tools and performance testing tools. Similarly, as shown in *Publication I*, the testing tools yielded results which indicated that the tools need configurability and extendibility, as several organisations also reported conducting test tool development themselves, not relying on the existing options.

4.2.3 Relation to the whole

Overall, this publication gives an insight into the test infrastructure and current state of software testing in the industry. The focus areas in this publication were on the applied tools and the purposes they are used for, discussing the automation tools in more detail. Other important observations in this publication concerned the test resources other than test tools, namely time restraints and human resources, and the types of testing methods applied in the test process.

The results of this study gave an insight into the amount of available resources in real-life organisations. The survey results indicated that the organisations do have access to a relatively high amount of test resources, as the average amount of resources was 70%[1], and that on average 27% of the project effort is spent on testing. These values are somewhat different than those which could be expected based on prior results from *Publication I.* On a larger scale, the results of this study also meant that the test tools

[1] for example, if organisation had 3 testers and they considered that they would need 4, this would translate to 75% of resources.

and test resourcing was generally at an acceptable level, and that the organisational management issues were more prominent than prior studies indicated. Furthermore, the average amount of effort allocated mainly to testing was less than expected, based on the software engineering literature (for example Kit 1995, Behforooz and Hudson 1996, Pfleeger and Atlee 2006).

4.3 Publication III: Analysis of the effects the applied development method has on the test process

4.3.1 Research objectives

The objective for this Grounded Theory study was to establish the relationship between the development process and the test process, and assess how the development method affects the practical implementation of testing.

4.3.2 Results

The results from this publication established several observations from test organisations. First and foremost was the observation that the development method itself is not a large influence on the way the testing is done, and that none of the development methods applied in the case organisations are inherently better or worse from the viewpoint of testing. In highly agile development, the approach allows more time for testing, as testing tasks can be started earlier than in traditional waterfall approach, although there are some difficulties in deployment of testing in the early iterations. By applying agile methods the resource requirements for testing were also more predictable. This can be considered an obvious advantage in organisations, where testing resources are limited and distributed competitively between different projects. In agile development, the customer participation or at least cooperation with the clients is one of the key aspects. Overall, the agile practices when compared against the traditional waterfall-development style changes the testing only in a few ways. The customer needs to understand the requirements and differences of the applied development method, the test strategy is focused on testing the new features and functionalities, and the organisation resource need is more predictable. As for problems in testing, the agile development may expose the organisation to problems with making and following the test plans. The different prominent categories are listed in Figure 10.

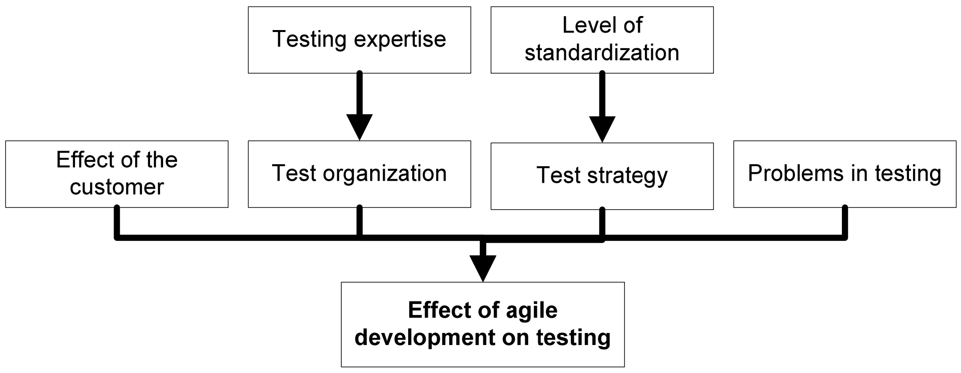


Figure 10: The aspects of agile development that affect the testing work

In general, the organisations which applied agile methods were also more flexible in terms of implementing and testing changes in the product. However, the agile approach also causes the development and testing to run in parallel, which is difficult to execute in practice and requires more coordination than traditional approach. From the viewpoint of strictly testing, agile methods offer some benefits such as early involvement or predictable resource needs, but also hinders testing in some areas, such as in availability and quality of documentation needed in the testing work, while making the test management more laborious.

4.3.3 Relation to the whole

This publication studied the effect the development process has on the test process, and concluded that the effect of development style is not very important from the viewpoint of test process activities. Even though changing the development process may change some process dynamics between development and testing, such as resource needs in different phases and customer participation, test process activities can be assessed separately from the development.

Additionally, the amount of agile development processes was relatively low. However, the study results indicated that even if the software organisations did not apply the entire agile development process, most of them had adopted some agile practices, such as code reviews, daily meetings or daily builds. Only few organisations considered agile practices to be completely unfeasible for their software process.

4.4 Publication IV: Analysis of the test case selection and test plan definition in test organisations

4.4.1 Research objectives

The objective of this Grounded Theory study was to observe and study the project level decision making in testing, and assess how the organisations decide on which test cases are included and which excluded from the test plan. The study also studied the prioritisation process of test cases, to establish if there were detectable patterns, which could explain the motivation behind the decisions.

4.4.2 Results

The study identified several components, which affect the decision making process and resulted to two stereotypical approaches on test case selection and prioritization method, named risk-based and design-based selection methods. The risk-based selection method was favoured in organisations, in which the test resources were limited or competed, and the decisions on test cases were made by testers themselves or designers in the lower levels of organisation. In design-based approach, the selection and prioritization process was done by the project-level management or dedicated expert. In the risk-based approach, the focus of testing was on verification, "what should be tested to minimise possible losses from faulty product", whereas the design-based approach focused on validation, "what should be done to ensure that the product does what it is supposed to do". More details are available in Table 9.

Overall, the study observed several testing-related components, which were tied to the test plan development. Such components as the test designers, the role of the customer, the resource availability, and the development approach seemed to have connection to the selected approach on test plan development. In addition, it was established that explorative testing (see Kaner et al. 1999), i.e. testing without a detailed case plan, was also connected to the test case selection approach: in many organisations where the test plan was design-based, doing test work without planned cases – "just using the system" – was considered an unproductive ad hoc approach.

Table 9. Two stereotypical approaches for test case selection

Category	Risk-based selection	Design-based selection
Test designers	Developers: programmers and testers	Managers: test and project managers
Development approach	Leans towards agile methods	Leans towards plan-driven methods
Testing resources	Limited	Sufficient
Explorative testing	Applied commonly	Applied rarely
Effect of policies in decisions on testing.	Small; most decisions done in project level.	Large; most decisions are based on company policies or customer requirements.
Customer influence	In the testing process	In the design process
Limitations of the model	Test case coverage may become limited.	Test process may become laborous to manage
Design concept	"What should be tested to ensure smallest losses if the product is faulty?"	"What should be tested to ensure that the product does what it is intended to do?"

However, test automation was observed to be rather independent from the test case selection approach. It seemed that the decision to apply test automation in testing work is based on other process factors, and test case selection or development of a test plan has only a little influence on it.

4.4.3 Relation to the whole

This publication focused on the project-level management process activities. The theme of the paper, the development of the test plan, and the prioritisation method of test cases studied, not only defined the process for the tested features, but also the low-level management of the testing work. The results indicated that all organisations have some systematic approach to deciding what should be tested and that in all observed organisations, some form of formal test management existed, even though in some cases the role of the test manager was not defined. In addition, the observations propose that the selection method for test cases and the fundamentals behind a test plan tend to steer towards two identified strategies; risk-based or design-based selections. With risk-based strategy, the main objective is to minimize the potential losses caused by missed errors; to use the available resources for maximum test coverage. With the design-based strategy, the objective is to validate the product; define an effective plan to cover the required test coverage.

4.5 Publication V: Analysis of the requirements for developing test process or adopting new testing methods in software organisations

4.5.1 Research objectives

In this qualitative Grounded Theory study, the focus was on establishing the requirements for organisation to start the test improvement process, and study how they adopt new testing techniques. An additional area of interest was to study how closely the ISO/IEC 29119 test process model (ISO/IEC 2010) fits the existing industry organisations.

4.5.2 Results

The main results of the study focus on the test process improvement in the software industry. The main observation of the study was that the organisations try to preserve the status quo, meaning that they do not develop their test process or try out new testing techniques unless the process is in dire need of changes. Even in organisations that continuously collect performance data and feedback from the test process, the results may be completely ignored if the existing process is "good enough". As process development exposes the organisation to a possibility of failure and unnecessary costs, the threshold for conducting process development is high, even if the applied change would be positive and sensible. Based on the collected data, a model was defined for this process, along with an explanation of the limitations of adopting new test techniques and the development of test processes in organisations. The model is illustrated in Figure 11.

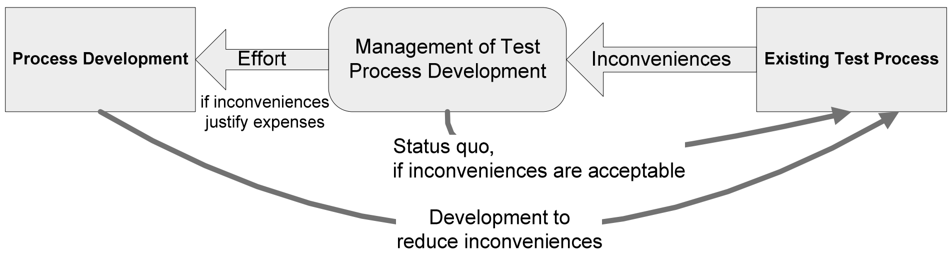

Process Development
Effort
if inconveniences justify expenses
Management of Test Process Development
Inconveniences
Existing Test Process
Status quo, if inconveniences are acceptable
Development to reduce inconveniences

Figure 11: Adopting new practices in test organisation.

The second topic of interest in this publication was the applicability and usability of the ISO/IEC 29119 test process in the real-life software organisation. The organisation representatives were asked to analyse how the proposed test process model differs from the approach the organisation is currently using and based on their experience and opinion, whether the model looks applicable or is it in need of changes. Based on the study results, the most common difference between the standard-defined test process model and the practical test process is in the number of organisational management and feedback processes. The interviewed organisations considered the model to have too many details in the upper management, and that the model itself lacked support for actually adopting the process model. However, the overall concept was considered feasible, not omitting any major components or important concepts.

4.5.3 Relation to the whole

This publication was the first that observed the entire software organisation, and instead of one aspect of testing work, such as, project management or test infrastructure, studied the organisation's behaviour. This study also explored the state of the test standard process model, finding some areas such as adaptability and the amount of details, which were considered difficult for the organisations. This study confirmed that the process model itself was feasible and did not omit anything obvious from the viewpoint of real-life software developers. On the development of the test processes, the study confirmed findings similar to those presented by Dybå (2008). Organisations prefer a status quo, and only conduct process development if the existing state becomes unbearable, even discarding the collected feedback data in a case where the process is at least in acceptable state. The organisations need a strong positive incentive to try out new techniques, even if the new method or proposed change in the way testing is done would seem sensible.

4.6 Publication VI: Analysis of associations between perceived software quality concepts and test process activities

4.6.1 Research objectives

The objective for this mixed method study, combining both the quantitative survey and qualitative Grounded Theory analysis, was to study the effect the quality-related aspects in software development and in software testing. In this study, the different quality characteristics as based on the ISO/IEC 25010 (ISO/IEC 2009) were tested in organisations, while different testing-related aspects such as outsourcing, open-source software in product and customer participation were studied from the viewpoint of perceived quality. A study by Garvin (1984) has identified the different types of software quality, and together with Jørgensen (1999) expressed a method of measuring the quality of a software product. In this study, these concepts were tested to see what types of quality are important to the software organisations.

4.6.2 Results

The central theme of this publication was in the different quality characteristics as defined in the ISO/IEC 25010 quality model, and studying how the perceived quality and different testing activities are related to each other. One of the most important observations of this publication was that almost all of the organisations do consider all of the different quality characteristics at least somewhat valuable to their product. The organisations were asked to evaluate how well the quality characteristic was taken into account in their product on a scale of 1-5[2]; the averages only differed from 3,3 to 4,2 between the answers. Moreover, organisations were allowed to give a score 0, "this characteristic is irrelevant to us", but this option was used only in 9 cases out of 248 assessments (3,6%), out of the 31 surveyed organisations. Because of these results, the original concept that organisations producing different types of products focus on different types of quality was rejected. Results for each quality characteristic are listed in Figure 12.

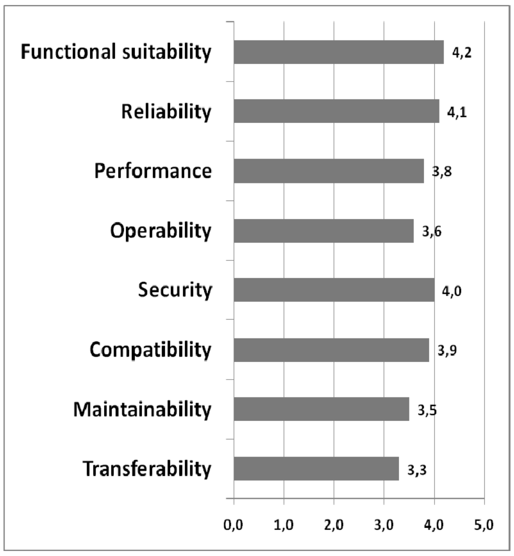


Figure 12: The realisation of different ISO/IEC 25010 quality characteristics.

The most important phase of a software process as a source of product quality was considered to be the development (average 4,3 on scale 1-5[3]), whereas the test process

[2] 1 = "this characteristic in our software is taken into account very badly", 5 = "this characteristic in our software is taken into account very well"

[3] 1 = "fully disagree, or this level is very bad in our organisation", 5 = "fully agree, or this level is very good in our organisation"

was considered less important (2,9). The results also indicated that within the organisations, the level in which the organisations already follow the concepts of the test process was somewhat low (3,3 on a scale of 1-5) in organisational activities, 3,4 on project level management and 3,5 on fundamental level activities. Overall, the most important factors in testing, which positively affected the perceived end-product quality were identified to be the trust between the software organisation and the clients, as well as existing process conformance with the concepts presented in the ISO/IEC 29119 test process standard, and finally the identification and communication of the desired quality characteristics throughout the software organisation. In addition, some concepts such as customer participation in product design and general control over the development project were identified to be somewhat important.

Besides the identification of testing aspects that affected the perceived end-product quality, the other important results were the aspects that were considered not to be very effective. Based on the survey and the qualitative analysis, such concepts as software criticality, product/service-orientation or outsourcing did not have a strong effect on the perceived end-product quality. Software criticality is obviously an important factor when deciding on how the product is produced, but changes in criticality do not alter the testing priorities or objectives of the testing work. Based on the results, it seems that the product domain is the most important factor affecting the selection of the tested components; software intended for Internet banking is generally tested for similar faults whether the target customer for the software is an individual users or a large corporations. Similarly, outsourcing was not considered a very important aspect affecting the perceived quality, in fact large organisations tended to think that outsourcing was helping the organisation to focus on the development of important features. It was also interesting to observe that the organisations which considered themselves to follow the principles of ISO/IEC 29119 test process model were also more confident that their product was of high, or at least acceptable, quality indicating a connection between the standard model and quality.

The last important observation was that the designers and testers rarely had similar ideas on the most important quality characteristics. In only two organisations were the same quality characteristics named and priorised in the same order between designers and testers. Overall, this confirms some of the concepts presented by Garvin (1984). The organisations do not have one clear image on the preferred quality, and attention should be paid to identifying and communicating the desired quality characteristics to all stakeholders.

4.6.3 Relation to the whole

This publication, along with *Publication II*, were the two studies where the survey data was used as the main research method, qualitative analysis providing additional viewpoints and validation for results which were considered important. In this publication, the test process was observed from the viewpoint of quality. Based on the survey results, it seems that development is considered a more important quality factor than testing. However, this can be somewhat expected, as the main objective for test process is in validating and verifying that the end-product is what was designed and works appropriately (Behforooz and Hudson 1996, Pfleeger and Atlee 2006), and quality can be understood in several contexts (Garvin 1984), one being that it "satisfies the user needs". If the user or customer satisfaction is not met in a design or development, the testing work cannot fulfil that type of quality. If the users do not think the product is of a high quality, it is difficult to argue that the product is actually of a high quality, for example, because of its technical excellence. In this context, it is plausible to say that the source of perceived quality is not in the test process, but in the design and development. However, the test process does have an influence on the end-product outcome and profitability (Huang and Boehm 2006, Tassey 2002). Therefore, it can be argued that the test process is used to realise the potential quality in the developed software.

4.7 Publication VII Self-assessment Framework for Finding Improvement Objectives with the ISO/IEC 29119 Test Standard

4.7.1 Research objectives

The objective of this study was to construct a maturity level-based framework to assess the existing test processes against the ISO/IEC 29119 standard process model and do preliminary testing on the validity and applicability of the framework.

4.7.2 Results

The concept of this publication was to combine the elements from a well-known and accepted software process evaluation model TIM (Ericson et al. 1997) to the draft of the ISO/IEC 29119 standard model to create a concept for a self-assessment framework. The objective was that the self-assessment framework could be applied to discover enhancement objectives in the organisational test process and alleviate the process adoption difficulties observed in the *Publication V*.

In the publication, a concept for combining the maturity levels from the Test Improvement Model and test processes of ISO/IEC 29119 was introduced. The different processes of the standard model (see Chapter 2.4) were assessed based on the maturity levels of TIM, which were customised to fit to the context of the processes:

- Level 0, Initial: The organisation does not have defined methods for this process activity.
- Level 1, Baseline: The organisation does have documented or at least generally agreed guidelines for these process activities, the process is systematically done to enable the finding and correcting of errors in the software.
- Level 2, Cost-effectiveness: The organisation tries to systematically promote cost-effectiveness or increase the efficiency of the process activities.
- Level 3, Risk-lowering: The organisation has metrics or other methods to enable the organisation to conduct risk-lowering and preventative actions in process activities.
- Level 4, Optimizing: The organisation has activities that aim to optimise the process; activities are done in a manner that is conceptually the same as in the standard.

The TIM model was applied as it was conceptually very similar to the standard; the key areas of TIM are assessed separately from each other, so that the organisation has a better understanding of what test process areas need most improvement. The evaluation work is easier to do as the number of simultaneously interacting concepts is kept reasonably small for an organisational assessment. Furthermore, the key areas of the TIM maturity model are similar to ISO/IEC 29119 processes; *the organisation* is conceptually close to organisational management process (OTP), *planning and tracking* to test management process (TMP) and TMCP, *test cases* to test plan process (TPP),

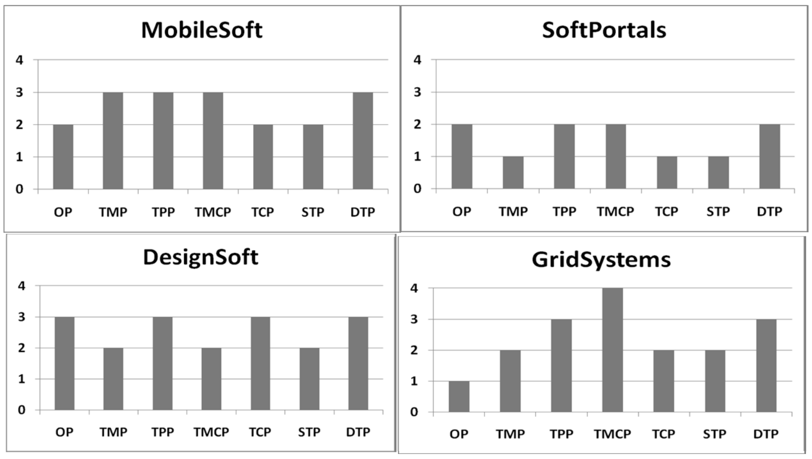


Figure 13: Assessment results using the experimental maturity levels

testware to STP and DTP, and *reviews* to TCP. Overall, four organisations from the fourth round interview organisations were selected for the pilot study, and assessed based on the interviews held during the research project. The resulting profiles are illustrated in Figure 13.

Besides the developed profiles, a number of practical enhancement proposals were derived based on the observations. The profiles were also tested out with three out of the four profiled organisations to assess the accuracy and development needs for the framework. The fourth case organisation had recently changed their process, so they declined to participate in this assessment. The main points of this feedback is presented in the Table 10, where "++" denotes very positive attitude towards the assessed attribute, and "– –" very negative.

Table 10. Feedback from the case organisations

	MobileSoft	**DesignSoft**	**SoftPortals**
Suitability of the framework	+; The applied approach is generally feasible.	++; Practical approach on quick and easy assessment of the level of different testing tasks.	+; Levels are too universal, but model itself seems to cover everything needed.
Suitability of the assessment levels	– –; In large organisation, the levels overlap, unnecessary processes for some organisations.	+; Usable, although some processes do not need to be better than cost-effective.	– ; Levels in general are OK but the definitions should be less ambiguous.
Accuracy of the profile	– ; The profile should be more detailed.	+; The profile was accurate enough, although with some differences.	++; The profile represents the organisation quite well.
Accuracy of the results	+; This type of feedback is always good for bringing out new ideas.	+; Results seemed usable.	++; Results same or similar to the internal discussions.
Framework development proposals	The assessment unit type and size should be clearly defined.	More definite descriptions for each framework level to reduce the overlap.	The assessment needs practical examples and more measurements.
Best profiler	An outsider from a third party, internal review is not accurate.	At least two manager-level employees; can be used internally.	A quality manager with a handpicked group of people, usable internally.

The overall attitude towards the developed framework was somewhat positive, although a few problems and enhancement proposals were identified. For example, the organisations considered that the framework profiling levels overlapped and were not very definite. Moreover, additional practical examples of actions denoting certain level of maturity were requested. However, the framework was applauded for the applicability as a tool for easy and fast assessment, and the accuracy of the results for being usable and similar to the issues discussed internally.

4.7.3 Relation to the whole

This publication discusses the test process from the viewpoint of process improvement. All of the results and ideas derived from the earlier studies, including the identified important test process components and applicability of the ISO/IEC 29119 standard, are applied in this publication in order to present a concept for an assessment tool to derive process improvement objectives. Obviously, the framework presented in this work is not complete, and it needs more studies before it can be established as a serious process improvement tool. However, the results and collected feedback from the proof-of-concept framework, so far at least, suggest that this type of self-assessment method could be feasible to implement based on the ISO/IEC 29119 test process model.

4.8 About the joint publications

For *Publication I,* the author analysed the previously collected interview data and wrote major parts of the publication.

For *Publication III,* the author participated in the writing of the publication, participated on the design and implementation of the data collection instruments, participated on the collection of the data, and acted as a supervisor for the Master's Thesis dissertation the publication is based on.

For *Publications II* and *IV-VII,* the author participated in the design and implementation of the data collection instruments, participated in the collection and analysis of the data, and wrote the major parts of the publications.

5 Implications of the results

The purpose of this chapter is to extract the research results from the different publications and present their implications as a conclusive summary. The original research problem and the objectives for the thesis remained same throughout the study. In this thesis, the overall research problem was approached by dividing the topic into several sub-questions, which were answered in the included publications. The objective of this thesis was to study the organisational test process, identify important factors affecting the test process, and define guidelines for the organisations in order for them to pursue better testing practices and develop their process towards operating methods presented in the ISO/IEC 29119 testing standard (ISO/IEC 2010). This objective was pursued by applying both qualitative and quantitative research methods when observing the organisations, and trying to understand how test processes work in real-life software development.

5.1 Implications for practice

The viewpoints of the thesis - organisational test processes and development of test processes - were selected based on the themes of the research project MASTO, which was a continuation project for the earlier project ANTI, focusing on the ISO/IEC 29119 testing standard and test processes of industrial organisations. In the preliminary phase of the study, a literature review on the topics and discussions with an expert group was used to understand the important factors of the study. Further concepts were derived from the earlier research project ANTI, from which the interviews regarding test process problems and enhancement proposals were used as a foundation for the data collection and analysis phase. The background work and analysis on test process problems based on the existing knowledge were reported in *Publication I*.

The assessment of different test strategy components was conducted in the second phase of the study, in the main data collection and analysis. In this phase, the components constituting the test strategy were divided to conceptual categories (see Figure 3), which were analysed in *Publications II-IV* and *VI*. In addition to these categories, an additional category of "Other" was also used based on the literature review suggestions and earlier phase results in order to study other possible areas of interest.

The first categories analysed were the testing tools and the testing personnel in *Publication II*. This publication studied the test resources in the organisations, focusing on identification of different types of test tools available in the organisations, the amount and types of test automation and human resources. Based on the results we were able to understand the situation of the testing work in the organisations, and identify what kind of approaches the different organisations use for testing software. The situation in industry was better than what could be expected based on the literature review; there were some organisations in which there still were problems with quality and availability of tools or testing resources in general. However, the average amount of 70 percent of the test resources, when compared with the organisation's self-defined optimum, was more than expected based on the prior knowledge established in *Publication I* and the ANTI-project results. This resourcing level also indicated that the issues of testing are more related to the organising and managing of the process itself, not on the availability of resources. It was also apparent that the most important knowledge for testers was the domain knowledge, which was mostly attained by working in the field. Additionally, even though the organisations had positive attitudes towards different certifications and standardisation programs, they are not very common in every-day application. Based on these results, it seems that the testing work in industry is in a better condition than could be expected based on the literature. It also implies that the management aspects in testing are more important than originally thought; in many organisations the resourcing was not an issue, but the test process still experienced problems, mostly in the early test phases such as integration or unit testing.

Publication III focused on organisational aspects and on the effect of the development method. The study results indicate that the production method has only limited effect on the test process itself. The end-product may be of high quality regardless of the applied production method, but based on the results it can be argued that successful application of the agile methods allows testing more time to work with the application in development and allows the organisation to be better prepared for test resource needs.

The application level of agile development processes was generally low, even though one organisation applied SCRUM principles in their development process. However, several organisations did apply some principles or practices which can be considered agile-oriented, such as daily builds, test automation, pair programming, code reviews or daily meetings, even if the amount of purely agile developers was limited. It was also apparent that agile development was favoured in patching and feature addition projects, whereas "traditional approaches" were favoured in main version development. This discussion was elaborated upon in *Publication VI*, in which the effect of open source resources was discussed. The conclusion was that the open source resources are useful when applicable in projects, but they do not offer significant benefits over the "closed" - bought - third party modules, mainly because the open source material has to be reviewed and tested before being accepted into the product. From the viewpoint of the developers, it was also apparent that the source of the code did not matter, as everything went through more or less the same procedures. Overall, the results from *Publication III* indicate that the development method does not necessarily affect the test process very much. The test work and development are separable entities, and how the development work is implemented, it may have only minor actual effect on how the testing work is organised. Based on these results, it could be argued that in studies focusing on the test process, the development process itself is not a major concern, provided that the development follows at least some credible approach.

Publication IV continued with the test process implementation, and observed the test process from the viewpoint of developing the test plan and selection of the test cases. The most interesting result in this publication was the strong division of the test plan development into two approaches, design-based and risk-based approaches. Based on the observations the organisations divided into two main design approaches, "What should be tested to ensure smallest losses if the product is faulty" and "What should be tested to ensure that the product does what it is intended to do". Stereotypically the risk-based approach was favoured when the amount of resources was limited and mainly the developers made the test planning decisions, whereas design-based approach was used mainly when the amount of resources was not a limiting factor and test planning decisions were affected by the customers and management. However, one important observation was that the project-level management does exist; in every organisation there was a person who was responsible for project-level test management.

Other observations include the application of test automation, which did not seem to follow the test plan pattern otherwise observed. Based on these results, it seems that the decision to apply test automation is not related to the applied approach on developing test plan. Another interesting finding was that the explorative testing was considered unprofessional and unproductive in several organisations. One possible explanation could be that the explorative testing is difficult to document, the results are difficult to predict and the effectiveness is dependent on the experience and professionalism of the tester doing the explorative testing. By applying these results in practice, the selection and prioritisation of applied test cases can be improved. Organisations should define what the test plan aims for, and based on that elaborate on the test plan development and selection of applied test cases. These results also confirm the existence of the project-level test management, indicating that the improvement activities focusing on test management can also improve the overall testing in projects.

The results of *Publication IV* can be used in organisations to develop the process of creating a test plan, and understand the weaknesses and needs of the different approaches. Basically, it seems that the objective of test process in project level is either to minimize the possible losses or make sure that the required features are acceptable. The results also indicate that at the project level, the test process activities are not always very formal, in many organisations, the designers and developers had a major influence on the testing decisions and even in large, well-resourced organisation some of the important test cases may be discarded if they are considered too resource-intensive. Furthermore, the role of the customer in development is not very active, usually the customer only approves the end-product in some form, not actively participating in the testing work.

The resulting end-product quality was observed in *Publication VI*. Based on the discussions of test resources and test tools in *Publication II* and the test plans in practice in *Publication IV*, this publication assessed the outcome by means of quality model as presented in the quality standard ISO/IEC 25010 (ISO/IEC 2009). The most important observation of this publication was the uniformity in the quality characteristics. The prior indication that different types of organisations would strongly focus on certain types of quality did not hold true in practice. In fact, most organisations did have at least some over all concern regarding the different quality characteristics, and even when assessing the practical implementation of the said characteristics in their products, the differences did not focus on any particular characteristic. An additional interesting result was that the software criticality and desired quality characteristics did not have a strong correlation; the desired quality comes from the product domain, and has only a weak relationship with the possible repercussions of the product. From the other quality-related concepts, customer participation, product/service-orientation and outsourcing also had only a weak correlation. The customer is an enabler for quality, but the customer has to either provide substantial amounts of resources or a commitment to have any effect on quality, and in large organisations, outsourcing was not considered to have any meaningful effect on the perceived end-product quality.

A further interesting finding was that organisations, which considered themselves to closely follow the concepts presented in the ISO/IEC 29119 test standard, also considered themselves to produce good quality. This result indicates that if the organisation has organised their testing work in a manner that has a systematic approach, including the different documents and feedback system, they are more confident about their work. Organisations that have a systematic or at least a codified approach on testing, also have objectives for their testing work, and tend to know the general level of quality they are pursuing. This would also imply that by introducing the ISO/IEC 29119 concepts into an organisation the perceived end-product quality would improve, and that communicating the preferred quality helps the test organisation to focus on the important characteristics. Even if the test process does not have a large influence on the origin of the quality, identifying and communicating the preferred quality characteristics in test organisation improves the perceived quality.

In *Publication V,* the focus of the studied topics shifted from the different influential test components to the test process itself. The main result of this publication was that the organisations do not actively pursue new techniques or ideas. In fact, organisations even discard the collected process feedback, if the process is "good enough". This status quo mentality can be explained by several factors. The process improvement process and introduction of new ideas costs money, and there are no guarantees that the improvements always justify the expenses, and the change resistance causes conflicts. Additionally, as observed by Dybå (2003), large organisations in particular have their own practices, which are based on their "history of success", which made them large to begin with. Against this history, the organisations have a certain institutionalized routines, which in stable situations can drive out the exploration for enhancements and better practices. Also Kaner et al. (2002) mention instances, where organisations apply, and during the project discard, overly ambitious test documentation practices which are dictated by their test strategy. Instead of developing their practices towards usefulness, these organisations blame the hindrances for causing oversights, and expect the strategy to work on their next project.

Other important result established in this publication was the feasibility assessment of the standard process model. In the earlier publications (*II-IV, VI*) the process components were discussed, but the overall model feasibility was open to interpretation. Based on the results, the model was feasible but had some criticism over limitations in adoptability and excess details. Overall, the observation that the organisations tend to resist process changes would indicate that the organisations are reactive in nature, they do process improvement but mostly to fix problems, not to improve outcomes. In practice, this would indicate that the organisations should identify the process problems earlier, and in order to enhance output they should try to implement process changes before absolutely necessary.

In *Publication VII,* the framework for assessing the test process against the ISO/IEC 29119 standard was introduced. Based on the prior results, this approach was considered appropriate, as it was established that the test processes are usually at least moderately resourced (*Publication II*), the development process does not excessively interfere with testing (*Publication III*), the project-level management exists in practice (*Publication IV*), there are aspects of quality which are affected by testing (*Publication VI*), and the overall model is feasible enough for application in a practical environment (*Publication V*). The framework was developed based on the concepts presented in the Test Improvement Model (TIM) (Ericson et al. 1997), by combining the TIM levels with the individual processes of the ISO/IEC 29119 model. This framework allowed organisation to assess how their current test process compared to the standard, and generated enhancement proposals to all levels of testing work, from organizational policies to fundamental test work at project-level. In practice, the objective was to create a light-weight process assessment tool, which could be used within an OU to uncover problems in the testing practices. Based on the feedback from organisations, the developed concept-level framework was a step towards a helpful tool, implying that there is a use for such a tool. The proof-of-concept framework can be seen as a one of the concepts from this study, which shows considerable potential for future research.

Overall, the major implications for the test process development in practice can be summarised into a number of major observations:

- The test process can be assessed and developed separately from the development process. The development method does not affect the test process activities to a large degree, as the development process creates a product, and the test process validates and verifies this product (*Publication III*).
- Besides resourcing, the test process hindrances and critical areas for development are also closely related to the organisational and project level management, an observation which was established in several of the publications (*II-VI*) in this dissertation.
- The concepts presented in the ISO/IEC 29119 test process model seem to enable better end-product quality, as the organisations, which had implemented test processes which followed the principles similar to the standard, were also more confident regarding their end-product quality (*Publication VI*).

- Even though the test process itself is not a major source of the perceived product quality, the best way for the test process to enhance the perceived end-product quality is to identify and communicate the preferred quality characteristics to all test organisation participants *(Publication VI*).

- Organisations are reactive, they perform process improvements in order to fix problems, not to improve outcome (*Publication V*). Organisations should identify the process problems earlier, and in order to avoid larger problems, try to implement process changes before they are absolutely necessary.

5.2 Implications for further research

The applied approach to the research problem enabled the study to identify a group of important components of testing and allowed the study to define a framework for self-assessment of test processes in organisations. Based on the existing knowledge on applicability of the Grounded Theory approach (Strauss & Corbin 1990, Seaman 1999) and the survey (Fink & Kosecoff 1985) applied, this study gained plausible results. It can be argued that this mixed method approach of combining the qualitative and quantitative data allowed us to observe the software organisations from the appropriate viewpoint and understand the dependencies of different testing concepts in a way which could have been difficult to do otherwise.

However, studying the entire software test process at several organisational levels quickly devolves into a huge cacophony of concepts. In this thesis, the scope was limited to observing the test process from the viewpoint of the test strategy and the organisation. More detailed concepts such as daily management of testing work, observing practical work in action, or assessment of the functional appropriateness of the applied testing tools should be analysed further, as they may indicate practical level problems related to the hypothesis and results presented in this work.

As well as approaching possible further research by examining the fundamental level of testing, another possible object of interest would be the self-assessment framework, which could be extended to a more serious assessment method, following the concepts of models such as TIM (Ericson et al. 1997) or MPTF (Karlstrom et al. 2005). To achieve this, more studies into organisations are needed to establish the thresholds for different assessments of process maturity levels, and in order to validate the model as a viable process development tool.

Overall, the major implications of this work for further test process research can be summarised in a few major observations:

- The organisations have difficulties in defining the objectives for test process development. The framework defined in *Publication VII* is a proof-of-concept for a tool that is lightweight enough to be used within an organisation unit, but is still applicable with a larger ISO/IEC 29119 model. It seems that there would be incentive for studying and developing these types of tools for lightweight test process improvement within one OU.

- In studies on test processes, the aspects from the development process are not very influential providing that the development follows some credible development method (*Publication III*). It would be a valid approach to conduct research focusing mainly on the process components of testing.

- More studies on test management at both an organisational and a project level should be conducted to study and address the management issues (*Publication II-VI*) of test processes.

6 Conclusions

In this chapter, the aim is to observe and discuss the research limitations and the results of this thesis. The entire research process and gained results were presented in the last chapters, so concerns regarding the applicability and usability of the generated results should be addressed. This thesis describes a study consisting of three separate but equally important phases; the preliminary phase, the main data collection and analysis, and the validation phase. Each of these phases studied the state of the software testing in a real-life organisation, offering a viewpoint on the practice and results on different aspects of testing work.

This thesis makes three main contributions. The first contribution is based on the results of the main data collection and analysis phase, in which the effect of different test process-related aspects to the actual test process were identified and studied from the viewpoints of both qualitative and quantitative data. The second contribution is the assessment of the ISO/IEC 29119 test process model (ISO/IEC 2010) in practical organisations. This work studied the model concepts and the applicability of the model from the organisational viewpoint, assessing the feasibility of the model in practical organisations, and highlighting improvement needs for the model. The third contribution is the analysis of the test processes as a whole, studying the process improvement process of test organisations, and identifying the process difficulties.

Based on the results presented in this dissertation, the test process is an entity, which can be assessed and improved separately from the overall development process. The observations and analysis on the test processes yielded the following hypotheses for application in both research and industrial contexts:

1. The test strategy establishes a framework for testing work at the project level. The following hypotheses promote the development of a test strategy to address the factors important for the testing work:

- The development of a test plan can be characterised as applying two stereotypical approaches. The first approach promotes a design-based approach, in which the testing work focuses on validating the object under testing. The second approach promotes the risk-based approach, where the testing work focuses on minimising the potential losses caused by the object under testing (*Publication IV*).

- There is only a limited association between development methods and test processes. The applied development method does not restrict the practical testing work to any large degree, or require compromises in the test process definitions (*Publication III*).

- The most important aspects in the test process which have positive association with the perceived end-product quality are trust between the customer and producer, a test process which conforms to the self-optimising processes as defined in the ISO/IEC 29119 standard and the communication of the preferred quality characteristics to all of the process stakeholders (*Publication VI*).

- In the test process resourcing, the organisations have an average of 70% of their self-defined "optimal" amount of resources and dedicate on average 27% of the total project effort to testing. Based on the study results presented in the literature and the survey data, the test process hindrances are also based on the efficiency factors and test management, in addition to simple resourcing issues (*Publication II*).

2. The ISO/IEC 29119 test standard is a feasible process model for a practical organisation with the following limitations as regards for applicability:

- The standard model can be characterised as being overly detailed in the definition of roles and activities. In the practical test organisations, the boundaries of different levels, processes and roles are less organised than the model presents (*Publication II, V, VI*).

- The process model is top heavy and places a considerable emphasis on the organisational aspects of the test process. Based on the qualitative analysis, the model defines several responsibilities for the upper management, many of which are performed, in real-life organisations, at the project-level management or at least not as systematically as defined in the model (*Publication V,VI*).

- The current standard model does not include a roadmap or phased process for adopting the model. This hinders the applicability of the model in organisations, as the organisations had difficulties in creating an approach which their existing process could adopt for the concepts presented in the standard (*Publication V, VII*).

3. The organisations do not actively try out new test methods and prefer a status quo in their test process. The following hypotheses relate to the test process development at the organisational level:

- The organisations do not test out new testing tools or apply new testing methods unless they have a significant external incentive to do so. Based on the qualitative analysis, these incentives are things like the current state of the existing process or business needs in the operating domain (*Publication V*).

- The organisational test process may have feedback processes in place to allow continuous development, but in practice, the organisations tend to disregard the evidence of process enhancement needs if the existing process still performs at least at an acceptable efficiency (*Publication V*).

- In test process development, the organisations need a way of relating their existing process to the proposed changes to understand the objectives, and more pragmatically, the requirements the process improvement needs to succeed (*Publication V*).

- The test process model presented in the ISO/IEC 29119 standard could be applicable as a foundation for test process assessment and development tool (*Publication VII*).

These conclusions are also listed in Table 11. In conclusion, the test process can be seen as an autonomous part of the development process, which can be assessed and developed separately from the actual development. In software organisations, the test process is related to several components, and by developing the test process, it is possible to enhance the perceived end-product quality and achieve better cost/efficiency ratios. However, in practice, the organisations tend to avoid process improvement, allowing the test process to exist in a state where it could be developed, but there is not a great enough incentive to start the process improvement process. To lower the threshold for process improvement, the organisations need practical information to understand and relate to the requirements and objectives. One approach to achieve this is to focus on the concepts highlighted in this study and compare the existing process with the ISO/IEC 29119 standard model by applying the conceptual framework introduced in this study. By continuing the qualitative research on the test processes, this framework could be extended to allow more details and better support for the organisations to develop their testing practices.

Table 11. The main contributions and implications of the thesis

Contribution	**Associated areas of test processes and testing work in general**
The development of a test plan can be characterised to two stereotypical approaches.	Test plan, test strategy.
There is only a limited association between development methods and test processes.	Test plan, test strategy, test policy
Besides test resources, the test process hindrances and areas for development are closely related to the organisational and project level management.	Test strategy, test plan, test completion reports
There are components in test process that have a direct effect on the perceived end-product quality.	Test process development, quality
The concepts presented in the ISO/IEC 29119 test process model enable better end-product quality.	Quality
The organisations have an average of 70% of their preferred test resources and dedicate on 27% of the total effort to testing.	Test resources, test plan
The ISO/IEC 29119 test standard is a feasible process model for a practical organisation with some limitations.	Test strategy
The standard model is too detailed in the definition of roles and activities.	Test strategy, Test process development
The standard model is top heavy and places a considerable emphasis on the organisational aspects of the test process.	Test policy, test strategy, test plan
The standard model does not include a roadmap or phased process for adopting the model, which hinders its applicability.	Test process development
The organisations do not actively try out new test methods and prefer a status quo in their test process.	Test process development
The organisations do not try new testing tools or testing methods unless they have a significant external incentive to do so.	Test resources, Test process development
The organisation may have feedback processes in place, but in practice, the evidence of process enhancement needs is disregarded.	Test completion reports, Test process development
In test process development, organisations need pragmatic requirements and distinct objectives.	Test process development
The ISO/IEC 29119 test standard is a feasible foundation for a test process development framework.	Test process development

6.1 Limitations of this thesis

All research projects have shortcomings, threats to their validity and limitation on the scope of their results, and this work is no exception. Overall, the first limitation of this study is the applied scope of the study. The scope of organisational test process restricts the study to the process concepts coming from the software organisation, and thereby not taking into account the possible process hindrances caused by the external stakeholders, such as upper management, public relations, marketing or sales. However, this limitation has to be accepted to allow the comparison between organisations of different sizes and operating domains, as the concept of an organisation unit (OU) is used to normalise the differences between observed units and allow meaningful comparison between organisation types.

Another limitation in the qualitative study is the sample organisation limitations. The observed organisations are of a high technical ability and professional software developers, meaning that the results may not reflect the problems of starting organisations or organisations, which rely on volunteers, as is common in open source communities. It is also possible to formulate more dimensions for defining new types of software development organisations, and by applying those new definitions to finding types of organisations, which were not covered in the theoretical sampling of this study. An additional concern was also that all organisations were Finnish or at least a Finnish subdivision of a larger multinational company. This concern would require additional focusing, as for example a study by (Borchers 2003, Wong and Hassan 2008) has observed that cultural differences have an impact on the software organisation and organisational culture as well. However, as with qualitative studies in general, the results of the study should be applied outside of its scope only as guidelines and recommendations. When applied in a new context, such as in the context of non-professional organisations, these results can be applied if the collected evidence suggests that there are enough similarities within the results of this study and the observations from the new context.

Overall, it can be argued that the number of organisations in this study is also rather low to allow a study of the effects of different concepts in the test process. Our objective was to establish the general effect of the different test process components and find the most important factors. It can be expected that by adding more organisations to the sample, the list of affecting factors would be more detailed, but the objective was not in compiling a comprehensive list of all possible variables, but to establish an understanding of the most important factors.

As for the quantitative survey sample, a sample of 31 organisations may also seem somewhat limited, but this limitation can also be avoided by designing the study to cater to these limitations. In our study, similarly to Iivari (1996), the sample size is small but sufficient if analysed correctly. In our study, the threat of overfitting the data was addressed by selecting the organisations to represent different software domains and types of organisations, and triangulating the data with different approaches. This approach was also used to defer a non-response bias in the results; by maintaining heterogeneity in the sample, the results do not favour certain types or sizes of software producers. Additionally, in a paper by Sackett (2001) there is discussion regarding conceptualisation of signal-to-noise-ratio in statistical research. Their approach is to define confidence as being based in the practicality of observations: confidence = (signal / noise) * square root of sample size. In practice, this indicates that the confidence for the result being non-random weakens if the amount of noise increases while the signal decreases. In the Sackett model, the attributes are abstracted, meaning that the noise can be considered to be the uncertainty of the answers or any source of variation. Even if this Sackett model is not mathematical but more probably a logically defined conceptualisation, the concept is that the confidence is strongly related to the noise in the survey data. In our study, the response rate was 74 % for the organisations originally considered for the survey, and the data collection in all sample organisations was conducted by the same researchers who also participated in the survey design to avoid misunderstandings of the questions and to obtain a larger response rate from the sample (Baruch 1999). In this sense, it can be argued that the noise-ratio in the survey data is low, allowing more confidence as to the appropriateness of the survey answers and the data in general. This confidence is important, as the study by Armstrong (2007) argues, in studies covering aspects from social sciences there may be problems with statistical approaches and validation of answers with a statistical approach.

As for the validity of the qualitative research, there are threats that should be addressed to assert the validity of the results (for example Denzin 1978, Robson 2002, Onwuegbuzie and Leech 2007). Golafshani (2003) discusses the validity and reliability of qualitative research, and makes some notations on the reliability and validity issues. The reliability and validity in a qualitative study are not the same as the traditional mathematically proved concepts. In the quantitative study, the reliability and validity are rather a conceptualisation of trustworthiness, rigor, and the quality of the study. To increase the validity in a qualitative study, the research must eliminate bias and remain truthful to the collected data and observed phenomena. Similar observations are also discussed by Moret et al. (2007). Moret et al. points out that qualitative and quantitative studies should not exclude, but rather complement each other. Each approach has their method of validating the data, and if the research question is quantitative by nature, it is appropriate to apply a quantitative approach to collect that part of the data even if the overall subject requires a qualitative analysis.

To guarantee the validity of the study we used probability sampling when selecting the OUs for the survey, and theoretical sampling when selecting the in-depth cases in the qualitative study. Robson (2002) lists three threats to validity in this kind of research: reactivity (the interference of the researcher's presence), researcher bias, and respondent bias and suggests strategies that reduce these threats. To avoid the researcher bias, the interviews were conducted by the researchers and for the data analysis, new researchers were brought in to participate in the data analysis to enable observer triangulation (Denzin 1978). During the data analysis for new publications, the new data (data triangulation) and the results were also compared with earlier quantitative and qualitative results (method triangulation) to further validate the study.

Overall, the mixed-method approach allows the study to validate the overall results with comparisons and cross-referencing between different sources of data from both qualitative and quantitative sources. In this sense, the threats to the validity of the results from this study are low, as the results can be traced through different analysis methods and are based on overlapping, but ultimately different data sets. Finally, as *Publications II-IV* and *VI* all observe the same dataset, which is also a subset for *Publications V* and *VII*, from several different viewpoints and test process areas, it can be argued that this dissertation itself presents the theory triangulation for the overall results.

6.2 Future research topics

The implications for future research were identified in the previous chapter. Nevertheless, the possible future research topics, which are based on this study, are numerous as the topic itself is based on an extensive background of testing-related concepts.

One possibility for a future research topic is an exploration of the implications of the new technology such as cloud computing or crowd sourcing in the application of testing. In this research, one of the objectives was to study and identify the important components of testing, which are important from the test process viewpoint (*Publications I-IV, VI)*. These publications cover several conventional aspects such as development methods, testing tools and test management, but the implications of changing software business and new technology along with distributed development methods such as cloud computing and crowd sourcing could be elaborated upon.

Another possible topic for future research could also be the development and extension of the self-assessment framework discussed in *Publication VII*. The proof-of-concept study has established that the framework could be an efficient tool for organisations and based on the literature related to the test process improvement these types of frameworks are useful in addressing the process objective issues of the organisations. By approaching the problem by combining concepts from related research areas such as organisational studies, group learning or psychology, with qualitative and comparative studies as regards existing process development models, the framework could be further developed towards a lightweight assessment model for application with the ISO/IEC 29119 test process standard.

References

Abrahamsson, P. (2001). “Commitment development in software process improvement: critical misconceptions”, Proceedings of the 23rd International Conference on Software Engineering, Toronto, Canada, pp. 71-80.

Abrahamsson, P., Salo, O., Ronkainen, J. andWarsta J. (2002). “Agile Software Development Methods: Review and Analysis”, VTT Publications 478.

Afzal, W. and Torkar, R. (2008), ' Incorporating Metrics in an Organizational Test Strategy’, IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW’08), 9.4.-11.4., Lillehammer, Norway.

Armstrong, J.S. (2007). "Significance tests harm progress in forecasting". International Journal of Forecasting 23: 321–327. doi: 10.1016/ j.ijforecast.2007.03.004

ATLAS.ti (2010). ATLAS.ti: The Qualitative Data Analysis Software, Scientific Software Development. Available at http://www.atlasti.com/, referenced 2.5.2010.

Bach, J. (1997). “Test automation snake oil”, 14th International Conference and Exposition on Testing Computer Software, Washington D.C., USA.

Baruch Y. (1999). "Response Rate in Academic Studies - A Comparative Analysis," Human Relations, vol. 52, pp. 421-438.

Behforooz, A. and Hudson F.J. (1996). “Software Engineering Fundamentals”, Oxford University Press, Oxford, NY, USA. ISBN: 0-19-510539-7

Becker, J. and Niehaves, B. (2007). “Epistemological perspectives on IS research: a framework for analysing and systematizing epistemological assumptions”, Information Systems Journal, 17: 197–214. doi: 10.1111/j.1365-2575.2007.00234.x

Bertolino, A. (2007). “Software testing research: achievements, challenges, dreams”. In International Conference on Software Engineering,” 2007 Future of Software Engineering, Minneapolis, MN, USA.

Boehm, B. (2002). “Get ready for Agile Methods, with Care”, Computer Vol. 35(1). DOI: 10.1109/2.976920

Boehm, B. and Turner, R. (2003), "Using Risk to Balance Agile and Plan-Driven Methods," Computer, vol. June, pp. 57-66.

Borchers, G. (2003). “The software engineering impacts of cultural factors on multi-cultural software development teams”, In Proceedings of the 25th International Conference on Software Engineering (ICSE '03). IEEE Computer Society, Washington, DC, USA, 540-545.

Briand L. and Labiche, Y. (2004). "Empirical studies of software testing techniques: challenges, practical strategies and future research", ACM SIGSOFT Software Engineering Notes, Vol. 29, Issue 5, pp. 1-3.

Brown, S. (2000). “Overview of IEC 61508. Design of electrical/electronic/ programmable electronic safety-related systems”. Computing & Control Engineering Journal, Vol. 11(1), pp. 6-12, doi: 10.1049/ccej:20000101

BSI (1998). “7925-1:1998 Software testing: Vocabulary”, British Standards Index.

Burnstein, I., Suwanassart, T., Carlson, R. (1996). “Developing a testing maturity model for software test process evaluation and improvement”, International Test Conference 1996 (ITC'96). doi: /10.1109/TEST.1996.557106

Carlsson, S.A. (2003). “Advancing Information Systems Evaluation (Research): A Critical Realist Approach”, Electronic Journal of Information Systems Evaluation, Vol. 6(2), pp. 11-20.

Chen, W. and Hirschheim, R. (2004). “A paradigmatic and methodological examination of Information systems research from 1991 to 2001”. Information Systems Journal, 14, 197–235.

Chen, Y., Probert, R.L. and Robeson, K. (2004), ‘Effective test metrics for test strategy evolution’, CASCON '04 Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research, pp. 111-123, 5.10.-7.10.2004, Markham, Canada.

CMMi Product Team (2010). “CMMI for Development, Version 1.3”, Software Engineering Institute, Carnegie Mellon University.

Conradi, R. and Fugetta, A. (2002) "Improving software process improvement", IEEE Software, Vol. 19, Issue 4.

Cronbach L. J. (1951), "Coefficient Alpha and the Internal Structure of Tests," Psychometrika, vol. 16, pp. 279-334.

De Grood, D.-J. (2008). “Generic Test Strategy”, Testgoal, Springer Berlin Heidelberg, pp. 119-123. ISBN 978-3-540-78828-7

Deemer, P., Benefield, G., Larman, C. and Vodde, B. (2010), “The Scrum primer”, version 1.2, Scrum Foundation.

Denzin, N. K. (1978), The research act: A theoretical introduction to sociological methods, McGraw-Hill.

Dybå, T. (2003). “Factors of software process improvement success in small and large organizations: an empirical study in the Scandinavian context”, Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering, pages 148-157, Helsinki, Finland. doi: 10.1145/940071.940092

EC (2003), "The new SME Definition User guide and model declaration" Enterprise and Industry Publications, European Commission, 2003.

Eisenhardt, K. M. (1989), 'Building Theories from Case Study Research', Academy of Management Review, vol. 14, no. 4, pp. 532-550.

Ericson, T., Subotic, A., Ursing. S. (1997): TIM - A Test Improvement Model, Software Testing, Verification & Reliability (STVR), vol. 7 (4), pp. 229-246, John Wiley & Sons, Inc..

Estublier, J., Leblang, D., Hoek van der, A. Conradi, R., Clemm, G., Tichy, W. and Wiborg-Weber, D. (2005). “Impact of software engineering research on the practice of software configuration management”, Transactions on Software Engineering and Methodology, Vol. 14(4). DOI: 10.1145/1101815.1101817

Farooq, A. and Dumke, R.R. (2007). “Research directions in verification & validation process improvement”, SIGSOFT Software Engineering Notes, Vol. 32(4). doi: 10.1145/1281421.1281425

Fink, A. (2003). “The Survey Handbook”, Second edition, SAGE Publications Inc. ISBN: 0-7619-2580-5

Fink, A. and Kosecoff, J. (1985). How to conduct surveys A Step-by-Step Guide. Newbury Park, CA: SAGE Publications, Inc..

Fowler, M. & Highsmith, J. (2001), “The Agile Manifesto”, Software Development, Aug. 2001, pp. 28-32.

Garvin, D.A. (1984). “What Does “Product Quality” Really Mean?”, Sloan Management Review, Issue 4, pages 25-43.

Geels, F.W. (2004), “From sectoral systems of innovation to socio-technical systems Insights about dynamics and change from sociology and institutional theory”, Research Policy, Vol. 33, pp. 897-920. doi:10.1016/j.respol.2004.01.015

Glaser, B.G. (2002), “Constuctivist Grounded Theory?”, Forum: Qualitative Social Research (FQS), Vol 3(3).

Glaser, B. and Strauss, A.L. (1967). The Discovery of Grounded Theory: Strategies for Qualitative Research. Chicago: Aldine.

Golafshani, N. (2003). “Understanding Reliability and Validity in Qualitative Research”, The Qualitative Report, Vol 8(4), December 2003, pages 596-607.

Hardgrave, B.C. and Armstrong, D.J. (2005). “Software process improvement: it's a journey, not a destination”, Communications of the ACM, Vol. 48(11), pages 93-96. DOI: 10.1145/1096000.1096028

Harrold, M.J. (2000). “Testing: a roadmap”, 22nd International Conference on Software Engineering, The Future of Software Engineering, pp. 61–72.

Highsmith, J. and Cockburn, A. (2001). Agile Software Development: the Business of Innovation. *Computer,* 34(9), 120-127. DOI: 10.1109/2.947100

Hirschheim, R. A. (1985), 'Information systems epistemology: an historical perspective', in R. H. E Mumford, G Fitzgerald, T Wood-Harper (ed.), Research Methods in Information Systems, North-Holland, Amsterdam.

Huang, L. and Boehm, B. (2006) How Much Software Quality Investment Is Enough: A Value-Based Approach, IEEE Software, Vol. 23(5), pp. 88-95, doi: 10.1109/MS.2006.127.

IEEE (1987). “ANSI/IEEE Std 1008-1987 IEEE Standard for Software Unit Testing”, IEEE Computer Society.

IEEE (2008). “829-2008 IEEE Standard for Software Test Documentation”, IEEE Computer Society, 2008. DOI: 10.1109/IEEESTD.2008.4578383

Iivari, J. (1996). "Why Are CASE Tools Not Used," Communications of the ACM, vol. 39, pp. 94-103.

ISO/IEC (2002). ISO/IEC 15504-1, Information Technology - Process Assessment - Part 1: Concepts and Vocabulary, 2002.

ISO/IEC (2005) ISO/IEC 25000, Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE, first edition, 2005.

ISO/IEC (2008). ISO/IEC 12207:2008 Systems and Software engineering - Software life cycle processes, 2008.

ISO/IEC (2009). ISO/IEC 25010-3, Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) - Quality Model, version 1.46, dated 28.1.2009.

ISO/IEC (2010). ISO/IEC WD 29119-2, Software and Systems Engineering— Software Testing .— Part 2: Test Process, version 2.1, dated 27.05.2010.

ISO/IEC/IEEE (2010) “24765:2010 Systems and software engineering - Vocabulary”, First edition, 15.12.2010, ISO/IEC and IEEE Computer Society. DOI: 10.1109/IEEESTD.2010.573383

ISTQB (2007). International Software Testing Qualifications Board (ISTQB), Certified Tester Foundation Level Syllabus, version 01.05.2007.

Jung, E. (2009). “A Test Process Improvement Model for Embedded Software Developments”, Proc. Of the 9th Internatinal Conference on Quality Software, 24.-25.8.2009, Jeju, South Korea.

Juristo, N. and Moreno, A.M. (2001). “Basics of Software Engineering Experimentation”, Kluwer Academic Publishers, Boston, USA. ISBN: 0-7923-7990-4

Juristo, N., Moreno, A.M. and Vegas, S. (2004). “Reviewing 25 years of testing technique experiments”, Empirical Software Engineering, Vol. 9(1-2), pages 7-44. DOI: 10.1023/B:EMSE.0000013513.48963.1b

Jørgensen, M. (1999). “Software quality measurement”, Advances in Engineering Software, Vol 30(2), pages 907-912.

Kaner C. and Bach J. (2005). “Black Box Software Testing”, Association for Software Testing.

Kaner, C., Bach, J. and Pettichord, B. (2002). “Lessons Learned in Software Testing: A Context-Driven Approach”, John Wiley & Sons Inc., New York. ISBN: 0-471-08112-4

Kaner, C., Falk, J. and Nguyen, H.Q. (1999). “Testing Computer Software”, second edition, John Wiley & Sons Inc., New York. ISBN: 0-471-35846-0

Karhu, K., Repo, T., Taipale, O. and Smolander, K. (2009). “Empirical Observations on Software Testing Automation”, Proceeding of the 2nd International Conference on Software Testing, Verification and Validation, Denver, CO, USA.

Karlström, D., Runeson, P., Nordén, S. (2005). “A minimal test practice framework for emerging software organizations”, Software Testing, Verification and Reliability (STVR), Vol. 15(3), pp. 145-166. DOI: 10.1002/stvr.317

Kautz,K., Hansen H.W. and Thaysen, K. (2000). “Applying and adjusting a software process improvement model in practice: the use of the IDEAL model in a small software enterprise”, Proceedings of the 22nd international conference on Software engineering, Limerick, Ireland, pages 626-633. doi: 10.1145/337180.337492

Kit, E. (1995), Software Testing in the Real World: Improving the Process, Addison-Wesley, Reading, MA.

Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., Emam, K.E. and Rosenberg, J. (2002). "Preliminary Guidelines for Empirical Research in Software Engineering," IEEE Transactions on Software Engineering, vol. 28, pp. 721-733.

Klein, H.K. and Myers, M.D.(1999). "A set of principles for conducting and evaluating interpretive field studies in information systems”, MIS Quarterly, vol. 23, pp. 67-94.

Koomen, T. and Pol, M. (1999). “Test Process Improvement: A practical step-by-step guide to structured testing”, Addison-Wesley, Great Britain. ISBN: 978-0-201-59624-3

Locke, K. (2001).”Grounded Theory in Management Research”, SAGE Publications, Thousand Oaks, CA, USA. ISBN: 0-7619-6427-4

Miles, M. B. and Huberman, A. M. (1994). Qualitative Data Analysis. SAGE Publications, Thousand Oaks, CA, USA.

Mingers, J. (2004), “Real-izing information systems: critical realism as an underpinning philosophy for information systems”, Information and Organization, Vol 14(2) April 2004, pp. 87-103. doi: 10.1016/j.infoandorg.2003.06.001

Moret, M., Reuzel, R., van der Wilt, G. J. and Grin, J.(2007). “Validity and Reliability of Qualitative Data Analysis: Interobserver Agreement in Reconstructing

Interpretative Frames", Field Methods, Vol. 19(1), pages 24-39. DOI: 10.1177/1525822X06295630

Myers, G.J. (2004), "The Art of Software Testing", 2nd edition, John Wiley & Sons, Inc., Hoboken, New Jersey, USA. ISBN: 0-471-46912-2

Ng, S.P., Murmane, T., Reed, K., Grant, D. and Chen, T.Y. (2004) 'A preliminary survey on software testing practices in Australia', Proc. 2004 Australian Software Engineering Conference (Melbourne, Australia), pp. 116-125.

Niekerk, van J.C. and Roode, J.D. (2009). "Glaserian and Straussian grounded theory: similar or completely different?", Proc. of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, DOI: 10.1145/1632149.1632163, Vanderbijlpark, South Africa.

Oh, H., Choi, B., Han, H., Wong, W.E. (2008). "Optimizing Test Process Action Plans by Blending Testing Maturity Model and Design of Experiments", Proc. of the 8th International Conference on Quality Software, pp. 57-66, doi: 10.1109/QSIC.2008.19 12.-13.8.2008, Oxford, UK.

Onwuegbuzie, A.J. and Leech, N.L. (2007). "Validity and Qualitative Research: An Oxymoron?", Quality and Quantity, Vol 41(2), pages 233-249. DOI: 10.1007/s11135-006-9000-3.

Paré, G. and Elam, J.J. (1997). Using Case Study Research to Build Theories of IT Implementation. The IFIP TC8 WG International Conference on Information Systems and Qualitative Research, Philadelphia, USA. Chapman & Hall.

Pather, S. and Remenyi, D. (2004), "Some of the philosophical issues underpinning research in information systems: from positivism to critical realism", SAICSIT '04 Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, Pretoria, South Africa, 2004.

Petschenik, N.H., (1985). "Practical Priorities in System Testing", IEEE Software, Vol. 2(5), pp. 18-23. DOI: 10.1109/MS.1985.231755

Pfleeger, S.L. (1999). "Albert Einstein and Empirical Software Engineering", Computer Vol. 32(19), pp. 32-38. DOI: 10.1109/2.796106

Pfleeger, S.L. and Atlee, J.M. (2006). "Software Engineering Theory and Practice", Third edition, Pearson International Edition, USA. ISBN: 0-13-198461-6

Pfleeger, S.L. and Kitchenham B.A. (2001)."Principles of survey research: part 1:turning lemons into lemonade", ACM SIGSOFT Software Engineering Notes, Vol. 26(6). DOI: 10.1145/505532.505535

Pino, F.J., Garcia, F. and Piattini, M. (2009). "Key processes to start software process improvement in small companies", Proceedings of the 2009 ACM symposium on Applied Computing, Honolulu, Hawaii, USA. DOI: 10.1145/1529282.1529389

Robson, C. (2002). "Real World Research", Second Edition. Blackwell Publishing.

Sackett, D.L. (2001). "Why randomized controlled trials fail but needn't: 2. Failure to employ physiological statistics, or the only formula a clinician-trialist is ever likely to need (or understand!)". CMAJ 165 (9): 1226–37, October.

Salvaneschi, P. and Piazzalunga, U. (2008). ”Engineering models and software quality modes: an example and a discussion”, Proc. 2008 International workshop on Models in software engineering, Leipzig, Germany, pages 39-44. DOI: 10.1145/1370731.1370741

Schmidt, R.C., (1997). “Managing Delphi surveys using nonparametric statistical techniques”, Decision Sciences Vol. 28, pp. 763-774.

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

Sjøberg, D.I.K. Dybå T. and Jorgensen, M. (2007). “The future of empirical methods in software engineering research”, International Conference on Software Engineering, 2007 Future of Software Engineering, Minneapolis, MN, USA.

Sjøberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.-K. and Rekdal, A.C. (2005). "A survey of controlled experiments in software engineering," *Software Engineering, IEEE Transactions on* , vol.31, no.9, pp. 733-753. DOI: 10.1109/TSE.2005.97

Slaughter, S.A., Harter, D.E. and Krishnan, M.S. (1998)."Evaluating the cost of software quality", Communications of the ACM, Vol. 41, Issue 8.

Soressi, E. (2010). “Introduction in safety rules EN954-1, EN13849 and EN62061”, 5th International Conference on System Safety 2010, 18.-20.10.2010, Manchester, UK, pp. 1-6. doi: 10.1049/cp.2010.0824

SPSS (2011). SPSS 17.0. Chicago: SPSS Inc. http://www.spss.com. Referenced 2.5.2011.

Strauss, A. and Corbin J. (1990). Basics of Qualitative Research: Grounded Theory Procedures and Techniques. SAGE Publications, Newbury Park, CA, USA.

Sulayman, M. and Mendes, E. (2010). “Quantitative assessments of key success factors in software process improvement for small and medium web companies”, Proceedings of the 2010 ACM Symposium on Applied Computing, Sierre, Switzerland, pages 2319-2323. DOI: 10.1145/1774088.1774568

Susman, G.I. and Evered, R.D. (1978). "An Assessment of the Scientific Merits of Action Research," *Administrative Science Quarterly*, (23), pp. 582-603.

Taipale, O. and Smolander, K. (2006). “Improving Software Testing by Observing Causes, Effects, and Associations from Practice”, the International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil.

Taipale, O., Smolander, K. and Kälviäinen, H. (2006a). “Cost Reduction and Quality Improvement in Software Testing”, Software Quality Management Conference, Southampton, UK.

Taipale, O., Smolander, K. and Kälviäinen, H. (2006b). “Factors Affecting Software Testing Time Schedule”, the Australian Software Engineering Conference, Sydney. IEEE Comput. Soc, Los Alamitos, CA, USA.

Taipale, O., Smolander, K. and Kälviäinen, H. (2006c). “A Survey on Software Testing”, 6th International SPICE Conference on Software Process Improvement and Capability dEtermination (SPICE'2006), Luxembourg.

Tan, M.T.K. and Hall, W. (2008). “Beyond Theoretical and Methodological Pluralism in Interpretive IS Research: The Example of Symbolic Interactionist Ethnography”, Communications of the Association of Information Systems, Vol. 19(1), article nbr. 26.

Tassey, G. (2002). The Economic Impacts of Inadequate Infrastructure for Software Testing. U.S. National Institute of Standards and Technology report, RTI Project Number 7007.011.

TMMi (2010). ” Test Maturity Model integration”, Version 3.1, TMMi Foundation, Ireland.

Utting, M. and Legeard, B. (2007). “Practical model-based testing: a tools approach”, Elsevier Inc.. ISBN: 978-0-12-372501-1

Weber, R. (2003) Editor’s comment: theoretically speaking. MIS Quarterly, 27, pp. 3-12.

Whitworth, B. (2006). “Socio-Technical Systems”, *Encyclopedia of Human Computer Interaction*, ed. Claude Ghaoui, pp. 559-566. Hershey PA: Idea Group Reference.

Wong, B. and Hasan, S. (2008). “Cultural influences and differences in software process improvement programs”, Proceedings of the 6th international workshop on software quality, Leipzig, Germany, pages 3-10. DOI: 10.1145/1370099.1370101

Appendix I: Publications

Publication I

Analysis of Problems in Testing Practices

Kasurinen, J., Taipale, O. and K. Smolander (2009), 16th Asia-Pacific Software Engineering Conference (APSEC), Batu Ferrenghi, Penang, Malaysia, 01.-03.12.2009. doi: /10.1109/APSEC.2009.17.

Analysis of Problems in Testing Practices

Jussi Kasurinen, Ossi Taipale and Kari Smolander
Department of Information Technology
Lappeenranta University of Technology
Lappeenranta, Finland
jussi.kasurinen | ossi.taipale | kari.smolander@lut.fi

Abstract—**The objective of this qualitative study was to explore and understand the problems of software testing in practice and find improvement proposals for these issues. The study focused on organizational units that develop and test technical software for automation or telecommunication domains, for which a survey of testing practices was conducted and 26 organizations were interviewed. From this sample, five organizations were further selected for an in-depth grounded theory case study. The analysis yielded hypotheses indicating that a software project design should promote testability as architectural attribute and apply specialized personnel to enhance testing implementation. Testing tools should also be selected based on usability and configurability criteria. These results of this study can be used in developing the efficiency of software testing and in development of the testing strategy for organization.**

Keywords-software testing, test process, problems, enhancement proposals, grounded theory

I. Introduction

Software testing is an essential part of software engineering and a central component which largely affects the final product quality [1, 2, 3, 4]. However, testing still has much potential to grow, as illustrated, for example, in the paper by Bertolino [1]. Bertolino discusses the future of testing and offers several objectives, like test-based modeling or completely automated testing, to aim for in the future. Similarly, Sjoberg et al. [5] also discuss the relevance of empirical research as a method for examining the usefulness of different software engineering activities in real-life situations. Additionally, Briand and Lapiche [6] also discuss empirical software engineering research from the viewpoint of applicability. Their opinion is that research on testing techniques should be tested in industrial settings. The human impact and experience are important factors in testing-related research, and therefore the most applicable results are gained by observing professional testing personnel [6].

A test process has practical limitations in resources, as well. A previous study by Taipale et al. [7] suggests that the main cost items for testing are personnel costs and automation costs. Process improvement, increased automation, and experience-related “know-how” were the major components in testing efficiency. The economic impact of improving testing infrastructure has also been discussed by Tassey [4] and Slaughter et al. [3], who established that process improvement increased software quality, and decreased the number of defects and subsequent testing and debugging costs.

Conradi and Fugetta [8] discuss software process improvement and maintain that the process improvement is not completely straight-forward model adoption. Even if there are process improvement models available such as ISO/IEC15504 [9] or CMMI [10], better results are achieved when the improvement is business- or user-oriented [8]. Therefore, the improvement process should be based on problems observed, and implement internal improvement proposals, not just acquire and adapt to external process models.

In this study, our specific objective is to identify the problematic areas of software testing in real-world organizations. We aim to understand better the difficulties that testing practitioners face, and based on that understanding, to derive hypotheses on how these difficulties or issues could be addressed. We do this by applying the grounded theory research method [11, 12] on observations made in real-life, industrial software producing organizations.

This study is conducted in accordance with the grounded theory research method introduced by Glaser and Strauss [11] and later extended by Strauss and Corbin [12]. Grounded theory was selected because of its ability to uncover issues from the practice under observation that may not have been identified in earlier literature [12]. This study is also a continuation for a series of studies in software testing in software business domain. These studies approach software testing practice empirically from various viewpoints, including process improvement [13, 14], schedules [15], outsourcing [7, 13], and test automation [16].

The paper is organized as follows: Firstly, we introduce the related research. Secondly, the research process and the grounded theory method are described in Section 3. The analysis results are presented in Section 4. Finally, the discussion and conclusions are given in Section 5.

II. Related Research

Software testing and software quality are discussed in several studies with many different approaches to process improvement [e.g. 2, 3, 4, 17, 18, 19]. Software quality improvement and increased testing efficiency are also central themes in trade literature [e.g. 20, 21]. There are also international standards such as ISO 9126 [22], which have been created to define software quality, although not without their own clarity issues [23].

Besides user-perceived quality, the software industry also has an economic incentive to commit to the development of better testing practices. For example, Tassey [4] reports the impact of inadequate testing infrastructure. This report discusses the effect of poor quality and insufficient testing practices in great detail, focusing on testing metrics, testing frameworks and improvement methods. Overall, the report indicates that in the U.S. alone, insufficient testing infrastructure causes annually 21.2 billion dollars worth of additional expenses to the software developers. From this estimate, 10.6 billion could be reduced with reasonable infrastructure and testing framework improvements [4]. The effect of the improvement of software quality as a cost reduction method is also discussed by other authors, such as Slaughter et al. [3] or Menzies and Hihn [19].

The research on quality enhancement and testing practices has also produced smaller, method-oriented studies to propose testing process enhancements. For example, Johnson [18] discusses the approach using technical reviews and formal inspections to enhance software quality. Johnson relates that although the technical review is a powerful tool, it is usually expensive and prone to human errors such as personality conflicts or ego-involvement. However, by adopting an operating environment and suitable support tools the technical review process is improved as the issues are addressed. The quality aspects are also an interest point for business themselves; some private corporations have documented their own improvement proposals and ways to enforce good testing practices and policies [e.g. 24]

Kelly and Oshana [2] have introduced statistical methods as a way to improve software quality. Their approach was able to increase the cost-effectiveness of software testing by applying a testing strategy to unit testing and by constructing usage models for each tested unit. If the usage model was appropriate, the number of errors found increased, resulting in better quality.

Cohen et al. [25] also noticed that the result of testing ultimately depends on the interpersonal interactions of the people producing the software. Capiluppi et al. [26] discuss the practice of outsourcing, which causes new requirements for testing and quality control, as knowledge on software systems and knowledge transfer with third parties affect the final quality.

As for the industry-wide studies on software testing, Ng, Murmane and Reed [27] have conducted a study on software testing practices in Australia. In the study, 65 companies located in major cities and metropolitan areas were surveyed for testing techniques, testing tools, standards and metrics. The central themes in this study were that the test process does not easily adopt new tools or techniques, as they are time-consuming to learn and master. A lack of expertise by practitioners was considered a major hindrance, as were the costs of adopting new techniques or training specialized testers.

Overall, there seems to be a multitude of approaches to control difficulties and gain benefits in software testing, ranging from technology-oriented tool introduction processes to observing and enhancing stakeholder interaction.

III. Research Process

Software testing practice is a complex organizational phenomenon with no established, comprehensive theories that could be tested with empirical observations [3]. Therefore, an exploratory and qualitative strategy following the grounded theory approach [11, 12] was considered suitable for discovering the basis of testing difficulties. According to Seaman [28], a grounded approach enables the identification of new theories and concepts, making it a valid choice for software engineering research, and consequently, appropriate for our research.

Our approach was in accordance with the grounded theory research method introduced by Glaser and Strauss [11] and later extended by Strauss and Corbin [12]. We applied the process of building a theory from case study research as described by Eisenhardt [29]. Principles for an interpretive field study were derived from [30] and [31].

A. Data Collection

The standard ISO/IEC 15504-1 [9] specifies an organizational unit (OU) as a part of an organization that is the subject of an assessment. An OU deploys one process or has a coherent process context, and operates within a set of business goals. An OU is typically a part of a larger organization, although a small organization may in its entirety be only one OU. The reason to use an OU as an assessment unit was that normalizing company size makes the direct comparison between different types of companies possible.

The population of the study consisted of OUs from small, nationally operating companies to large internationally operating corporations, covering different types of software manufacturers from hardware producers to contract testing services.

For the first interview round, the selection from the population to the sample was based on probability sampling. The population was identified with the help of authorities, and the actual selection was done with random selection from the candidate pool. For the first round, a sample of 26 OUs was selected. From this group, five OUs were further selected as the case OUs for the second, third and fourth interview rounds. These five cases were selected based on the theoretical sampling [31] to provide examples of polar types of software businesses [29]. These selected cases represented different types of OUs, e.g. different lines of business, different sizes and different kinds of operation, enabling further rounds in order to approach the test process concepts from several perspectives. Managers of development and testing, testers, and systems analysts were selected as interviewees because these stakeholders face the daily problems of software testing and are most likely able to come up with practical improvement proposals. The interviews lasted approximately one hour, and were conducted by two researchers to avoid researcher bias [32, 33]. The OUs and interviewees are described in Table 1.

TABLE I. OUs AND INTERVIEWEES

Interview round(s)	OU	Business	Company size[1]	Interviewees
First	All 26 OUs, cases included	Automation or telecommunication domain, products represented 48.4% of the turnover and services 51.6%.	OUs from large companies (53%) and small/medium-sized enterprises (47%). The average size for OU was 75 persons.	Managers; 28% were responsible for testing, and 20% were responsible for development, 52% both.
2nd, 3rd and 4th	Case A	A MES producer and integrator	Large/international	Testing manager, tester, systems analyst
2nd, 3rd and 4th	Case B	Software producer and testing service provider	Small/national	Testing manager, tester, systems analyst
2nd, 3rd and 4th	Case C	A process automation and information management provider	Large/international	Testing manager, tester, systems analyst
2nd, 3rd and 4th	Case D	Electronics manufacturer	Large/international	Testing manager, 2 testers, systems analyst
2nd, 3rd and 4th	Case E	Testing service provider	Small/national	Testing manager, tester, systems analyst

[1]SME definition [34]

The first interview round contained both structured and theme-based questions. The objective was to understand the basic practices in testing, identify process problems, and collect improvement proposals. The interviewees were managers of development or testing, or both. The questions of the first round concerned general information regarding the OU, software processes and testing practices, and the development environment of the OU. The interviewees of the second round were managers of testing, those of the third round were actual testers, and in the fourth round they were systems analysts. The objective of these interview rounds was to achieve a deeper understanding of software testing practice from different viewpoints, and further elaborate on the testing process difficulties. The questions reflected this objective, being theme-based and focusing on the aspects of testing such as the use of software components, the influence of the business orientation, knowledge transfer, tools, organization and resources.

Before proceeding to the next interview round, all interviews were transcribed and analyzed for new ideas to emerge during the data analysis. The new ideas were then reflected in the following interview rounds.

B. Data Analysis

The grounded theory method contains three data analysis steps: open coding, where categories of the study are extracted from the data; axial coding, where connections between the categories are identified; and selective coding, where the core category is identified and described [12]. First, the prior data was analyzed to focus on the issues in the later interview rounds. The categories and their relationships were derived from the data to group concepts pertaining to the same phenomena into categories.

The objective of the open coding was to classify the data into categories and identify leads in the data. The process started with "seed categories" [35] that contained essential stakeholders and known phenomena based on the literature.

Seaman [28] notes that the initial set of codes (seed categories) comes from the goals of the study, the research questions, and predefined variables of interest. In the open coding, new categories appeared and existing categories were merged because of new information that surfaced in the coding. At the end of the open coding, the number of codes exceeded 196 and the codes were grouped to 12 categories.

The objective of second phase, the axial coding, was to further develop separate categories by looking for causal conditions or any kinds of connections between the categories.

The third phase of grounded analysis, the selective coding, was used to identify the core category [12] and relate it systematically to the other categories. As based on [12], the core category is sometimes one of the existing categories, and at other times no single category is broad or influential enough to cover the central phenomenon. In this study, the examination of the core category resulted in a set of software testing concepts, categorized into lists of issues coupled with improvement proposals.

IV. ANALYSIS RESULTS

In the categorization, the factors that caused the most problems in software testing and resulted in the most improvement ideas were identified from the research data, grouped, and named. We developed the categories further by focusing on the factors that resulted in or explained the problems or improvement ideas, while abandoning categories that did not seem to have an influence on the testing activities. The categories are listed in Table 2.

A. Developed Categories

Overall, the categories were developed to describe the common themes of test process observations of different OUs. The categories either described process difficulties in the organization or proposed enhancement over existing procedure. In some cases, the category-related topic did caused problems in the test process, but the organization did

TABLE II. CATEGORIES FOR THE CASE OUS

Category	Description
Testing tools	Attributes associated with testing tools, for example availability, usability, and upkeep.
Testing automation	Testing automation-related issues and improvement proposals.
Knowledge transfer between stakeholders	Issues related to knowledge transfer between stakeholders in the software development organization.
Product design	Development and testing issues related to, for example, the product architecture, feature definition, and design.
Testing strategy and planning	Problems and improvement proposals related to, for example, the testing strategy, resource allocation, test case planning, and test case management.
Testing personnel	Issues related to the testing personnel and personal expertise.
Testing resources	Issues related to the availability or amount of resources allocated to testing.

not offer any solution or enhancement proposal to correct this situation. Similarly, in some cases the category topic had enhancement proposal without actually being perceived as an actual process problem.

The category "testing tools" described the quality attributes of the tools, for example, availability and usability. Related problems, the complexity of using the tools, as well as the design errors in the tools and in the user interfaces were included in this category. Also improvement proposals including tool requests or application suggestions were included in this category.

The category "testing automation" described problems from any level or type of testing automation. For the improvement proposals, the application areas, ways to increase testing effectiveness with automation or cost-saving proposals for existing test automation were included.

The category "knowledge transfer between stakeholders" described the problems and improvement proposals for knowledge transfer and -sharing within the OU and between clients or third party participants.

The category "product design" described the shortcomings and possibilities related to the product architecture, feature definition or design phase-related testing problems, such as unclear features or late architectural revisions.

The category "testing strategy and planning" described the barriers caused by the lack of a testing strategy or test case planning. This category also incorporated issues caused by the testing priorities and the relationship between product development and product testing.

The category "testing personnel" described the staff-related problems and improvement proposals for personnel-related issues. These included, for example, the lack of expertise in testing, unavailability of in-house knowledge or purely a lack of human resources.

Finally, the category "testing resources" described the resource-related issues as in the availability of tools, funding or time to complete test phases.

B. Observations and Hypotheses

Based on the categories defined from the observed problems in software testing practice, we formed hypotheses to explain the process difficulties or summarize the observed phenomena. The hypotheses were shaped according to the analysis and categorized observations, which are presented in Table 3.

The line of business or company type did not seem to have a major influence on the fundamental difficulties encountered in the test process, further indicating that the OU-level observations are useful in analyzing testing organizations. However, there were some concerns and differences which were caused by the upper, corporate level, differences. Large companies with separate testing facilities and in-house developed testing tools seemed to be more vulnerable to testing tool errors, and had to use their resources in maintenance of their testing tools. Additionally, these companies usually had requirements to direct project resources to comply with design standards or offer legacy support. Small businesses were more vulnerable to resource limitations and they had to optimize their resources more carefully to minimize redundancy and overheads. For example, testing tool development as an in-house production was an unfeasibly large investment to a small company. However, smaller companies benefited from personal level knowledge transfer and had more freedoms in adjusting testing strategy to respond to the project realities, as there were fewer corporate policies to comply and follow.

In the following, we will present the results of our analysis in the form of a condensed list of hypotheses.

1) Hypothesis 1: Product design for testability should be a focus area in architectural design. In all case OUs, the test process could be enhanced by taking the test design into account during product planning. Systematic architecture, clearly defined feature sets, and early test personnel participation should contribute to ease the test planning and to achieve a better test coverage or savings in the project budget.

"*Yes it [test planning] has an effect [on the project], and in fact, we try to influence it even at the beginning of the product definition phases, so that we can plan ahead and create rhythm for the testing.*" – Testing manager, Case C.

"*The cost effectiveness for [errors] found is at its best if, in the best case, the errors can be identified in the definition phase.*" – Test Manager, Case E

Case C also reported that by standardizing the system architecture they could more easily increase the amount of testing automation in the software process.

"*When the environment is standardized, the automation can be a much more powerful tool for us.*" – Tester, Case C

2) *Hypothesis 2: The testing processes need to clearly define the required resources and separate them from the other project resources.* This condition persists in all of the case OUs. In general, the large companies had a tendency to cut testing to meet the deadline, whereas the small companies either worked overtime or scaled down the test coverage. On two OUs, the development process was even allowed to use testing resources if it was running overtime. All case OUs reported that they needed better testing strategy or plan more to help resource allocation.

"*And of course these schedules are tight, and it may be that the time left for testing is not sufficient, but we can pretty much manage these issues because of the early planning.*" – Tester, Case A.

In some OUs, the testing department had a limited option to change the product deadlines to allow more testing. Some OUs also expressed that the testing team should be able to send the product back to development if certain minimum criteria are not met.

"*We should not do redundant work. If the release is of*

TABLE III. PROBLEMS AND ENHANCEMENT PROPOSALS OF THE CASE OUS

Case OU Category		Case A	Case B	Case C	Case D	Case E
Testing tools	***Problem***	Complicated tools cause errors.	Commercial tools have limited usability	Complicated tools cause errors.	Complicated tools cause errors.	Commercial tools have limited usability
	Enhancement proposal	-	Over-investmentshould be avoided.	Error database to observe test process.	Less error-prone testing tools	Multiple tools eliminate tool bias in results.
Testing automation	***Problem***	-	Reliability issues cause expenses.	Unused, no suitable personnel.	-	High prices limit the automation applicability.
	Enhancement proposal	Dedicated testers to use automation	Component compatibility tests should be automated.	-	Test report automation	Automate regression testing
Knowledge transfer between stakeholders	***Problem***	Outdated, unusable documentation.	Too little communication.	Misunderstandings between testing and development.	Redundant investments.	Deficient product testing.
	Enhancement proposal	Developers to participate in testing team meetings	Promote communication between teams	Promote communication between teams	Results available to all project participants.	Dedicated people for inter-team communication.
Product design	***Problem***	Tailored features cause additional testing.	Feature development uses testing resources.	Support for legacy systems restrict design.	-	Product design uses resources from testing.
	Enhancement proposal	Design should promote testability.	Design should promote testability.	Systematic architecture would help testing.	Design should promote testability.	Systematic architecture would help testing.
Testing strategy and planning	***Problem***	The last test phases are scaled down if necessary.	Testing is done on overtime if necessary.	The last test phases are scaled down if necessary.	Testing has a guaranteed but fixed timetable.	Lack of resources causes scaling down testing.
	Enhancement proposal	Testing strategy to minimize the time-related risks.	Testing strategy to prevent unnecessary testing.	Features frozen after set deadline to help test planning.	Testing strategy to help focus on critical test cases.	Testing strategy to help prevent unnecessary testing.
Testing personnel	***Problem***	-	-	-	Carelessness, "battle fatique" in testing work.	Expertise is expensive to acquire.
	Enhancement proposal	Specialized testers would enhance the process.	Testers should work in pairs with designers.	-	Separate teams for system and integration tests.	Specialized testers would enhance total process.
Testing resources	***Problem***	Low product volume limits resource investments.	-	Not enough time for thorough testing.	Infrastructure costs limit testing.	Lack of testing environments limit test process.
	Enhancement proposal	Own test laboratory to speed up the process.	-	Bug-tracing process for testing tools.	-	Own test laboratory to speed up the process.

such a poor quality that it should not be tested [for release] at all, do not have the option of sending it back to development for additional debugging." –System Analyst, Case E

Different from the other, Case D reported that they have a guaranteed, fixed time allocation for testing. In this case, the testing strategy was used to optimize the testing process to cover the critical areas.

"[Based on the testing strategy] if there is a product with new features with specifications that need to be tested, we can be sure that it is tested and verified before the product is built." –Tester, Case D

3) Hypothesis 3: The selection of testing tools and testing automation should focus on the usability and configurability of possible tools. All large companies reported that the false positives due to complicated or faulty testing tools cause additional resource losses, whereas small companies related that the limited technology support for commercial tools restricted their test processes.

"*When an error is found, we should immediately establish whether or not the error was in the testing environment. This could save us one unnecessary working stage.*" – System Analyst, Case D

The interviewees thought widely that automation tools were error-prone or costly, but also that they could be used to automate recurring test cases.

"*We should automate as many areas as possible, but then we should also have people to create testing automation...*" – System Analyst, Case A

4) Hypothesis 4: Testing should be executed by specialized personnel. Specialized testers seemed to make the overall testing phase more efficient by enabling faster reactions to encountered problems. Both small company-based OUs reported that they would benefit from creating a completely independent testing laboratory, but were unable to do so because of resource restrictions.

"*We should keep everything in our own hands. The extreme scenario where we have private testing environments, however, is too much for us now.*" – System Analyst, Case B

The large company-OUs proposed additional human resources to eliminate testing errors and focus on creating parallel testing phases for product modules.

"*...it would be optimal if simultaneously, while software engineers figure out why [the error took place], we could verify that they are not caused by the testing environment.*" – System Analyst, Case D

V. Discussion and Conclusions

The objective of this study was to understand the problems of software testing and based on this understanding, to develop hypotheses of how the testing practice should be improved.

We observed that testing personnel issues, test process and strategy issues were rather independent from the business orientation or the company size. Therefore the OU-level comparison of different types of software companies, such as in this study, can be used to observe, compare, and develop internal activities such as testing.

In our study, four main hypotheses were derived. The first hypothesis emphasizes testability as an architectural design objective. This is easily bypassed, which leads to slower and more complicated testing processes. This problem may well be the root cause for the second hypothesis, the requirement of a well-defined test plan and realistically allocated resources for the test infrastructure.

The third hypothesis, the maintainability and usability of testing tools is one of the reasons why test resources should be separated from the development resources. For several organizations ill-suited or defective test tools were causing the test organization to waste time on manual confirmation of the tool results, which was practically a redundant task. Similarly the fourth hypothesis, the requirement of separate testers in the organization is understandable. It could be a waste of resources to use developers to do the testing work, which could be done more efficiently by dedicated testing personnel.

These basic findings seem to be in line with a similar, earlier study by Ng, Murmane and Reed [27]. Their study concluded that time restraints prevent new techniques from being introduced and that expenses hinder test process improvement. Our results imply the same, and indicate that testing tool applicability and their general usability are major additional factors for testing efficiency. These phenomena, testing tool applicability, time constraints and personnel expertise, also seem to be general problems, because they were found in all types of case OUs in our study, regardless of business area, available resources or company size.

Our analysis suggests that testing organizations do not gain any special benefits from belonging to a large organization. All case OUs reported that allocated time was the first issue in testing, restricting the test process to cover only the bare essentials. All of the organizations had also recognized that their products needed better testability, and that their resource base was too limited. It is plausible to believe that most of these issues are best handled by designing a better test strategy, including, for example, testing- and resourcing plans.

The limitation of this study is the number of case OUs. It is obvious that increasing the number of cases could reveal more details. However, our target was not to create a comprehensive list of issues that affect testing organizations, but to increase understanding of problems in testing practices by covering important factors from the viewpoint of our case OUs.

We believe that paying more attention to the known fundamental issues when organizing testing - selecting a testing strategy, planning and reserving testing resources - the efficiency and results of testing can be improved significantly. The results of this study can be used in the development of testing organizations and generally to avoid common pitfalls.

Acknowledgment

This study was supported by the ANTI project (www.it.lut.fi/project/anti) and by the ESPA project (http://www.soberit.hut.fi/espa/), both funded by the Finnish

Funding Agency for Technology and Innovation, and by the companies mentioned in the project pages.

REFERENCES

[1] A. Bertolino, “Softare testing research: achievements, challenges, dreams”. In International Conference on Software Engineering,” 2007 Future of Software Engineering, Minneapolis, MN, USA, 2007.

[2] D.P. Kelly and R.S. Oshana, "Improving software quality using statistical testing techniques", Information and Software Technology, Vol 42, Issue 12, 2000.

[3] S.A. Slaughter, D.E. Harter and M.S. Krishnan, "Evaluating the cost of software quality", Communications of the ACM, Vol. 41, Issue 8, 1998.

[4] G. Tassey, “The Economic impacts of inadequate infrastructure for software testing”, U.S. National Institute of Standards and Technology report, RTI Project Number 7007.011, 2002

[5] D.I.K. Sjoberg, T. Dybå and M. Jorgensen, “The future of empirical methods in software engineering research”, International Conference on Software Engineering, 2007 Future of Software Engineering, Minneapolis, MN, USA, 2007.

[6] L. Briand and Y. Labiche, "Empirical studies of software testing techniques: challenges, practical strategies and future research", ACM SIGSOFT Software Engineering Notes, Vol. 29, Issue 5, pp. 1-3, 2004.

[7] O. Taipale, K. Smolander and H. Kälviäinen, “Cost reduction and quality improvement in software testing”, Software Quality Management Conference, Southampton, UK, 2006.

[8] R. Conradi and A. Fugetta, "Improving software process improvement", IEEE Software, Vol. 19, Issue 4, 2002.

[9] ISO/IEC, ISO/IEC 15504-1, Information Technology - Process Assessment - Part 1: Concepts and Vocabulary, 2002.

[10] Capability Maturity Model Intergration (CMMI), version 1.2, Carnegie Mellon Software Engineering Institute, 2006.

[11] B. Glaser and A.L. Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research. Chicago: Aldine, 1967.

[12] A. Strauss and J. Corbin, Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Newbury Park, CA: SAGE Publications, 1990.

[13] O. Taipale and K. Smolander, “Improving software testing by observing practice”, International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, 2006.

[14] O. Taipale, K. Smolander and H. Kälviäinen, “A survey on software testing”, 6th International SPICE Conference on Software Process Improvement and Capability dEtermination (SPICE'2006), Luxembourg, 2006.

[15] O. Taipale, K. Smolander and H. Kälviäinen, “Factors affecting software testing time schedule”, the Australian Software Engineering Conference, Sydney, Australia, 2006.

[16] K. Karhu, T. Repo, O. Taipale and K. Smolander, “Empirical observations on software testing automation”, IEEE Int. Conf. on Software Testing Verification and Validation, Denver, USA, 2009.

[17] V.R. Basili and R.W. Selby, "Comparing the effectiveness of software testing strategies", IEEE Transactions on Software Engineering, Vol. SE-13, Issue 12, 1987.

[18] P.M. Johnson, H. Kou, M. Paulding, Q. Zhang, A. Kagawa, and T. Yamashita, "Improving software development management through software project telemetry", IEEE Software, Vol. 22, Issue 4, 2005.

[19] M. Menzies and J. Hihn, “Evidence-based cost estimation for better-quality software”, IEEE Software, Vol. 23, Issue 4, pp. 64-66, 2006.

[20] E. Dustin, Effective Software Testing: 50 Specific Ways to Improve Your Testing, Addison-Wesley Professional., 2002.

[21] G.J. Myers, The Art of Software Testing, 2nd Edition. John Wiley & Sons, Inc, 2004.

[22] ISO/IEC, ISO/IEC 9126-1, Software engineering – Product quality - Part 1: Quality model, 2001

[23] H.-W. Jung, S.-G. Kim and C.-S. Chung, “Measuring software product quality: a survey of ISO/IEC 9126”. IEEE Software, Vol. 21, Issue 5, pp. 88-92, 2004.

[24] R. Chillarege, “Software testing best practices”, Technical Report RC21457. IBM Research, 1999.

[25] C.F. Cohen, S.J. Birkin, M.J. Garfield and H.W. Webb, "Managing conflict in software testing," Communications of the ACM, vol. 47, 2004.

[26] A. Capiluppi, J. Millen and C. Boldyreff, “How outsourcing affects the quality of mission critical software”, 13th Working Conference on Reverse Engineering, Benevento, Italy, 2006.

[27] S.P. Ng, T. Murmane, K. Reed, D. Grant and T.Y. Chen, “A preliminary survey on software testing practices in Australia”, Proc. 2004 Australian Software Engineering Conference (Melbourne, Australia), pp. 116-125, 2004

[28] C.B. Seaman, "Qualitative methods in empirical studies of software engineering", IEEE Transactions on Software Engineering, vol. 25, pp. 557-572, 1999.

[29] K.M. Eisenhardt, "Building theories from case study research", Academy of Management Review, vol. 14, pp. 532-550, 1989.

[30] H.K. Klein and M.D. Myers, "A set of principles for conducting and evaluating interpretive field studies in information systems", MIS Quarterly, vol. 23, pp. 67-94, 1999.

[31] G. Pare´ and J.J. Elam, “Using case study research to build theories of IT Implementation”, IFIP TC8 WG International Conference on Information Systems and Qualitative Research, Philadelphia, USA, 1997.

[32] N.K. Denzin, The research act: A theoretical introduction to sociological methods. McGraw-Hill, 1978.

[33] C. Robson, Real World Research, Second Edition. Blackwell Publishing, 2002.

[34] EU, "SME Definition," European Commission, 2003.

[35] M.B. Miles and A.M. Huberman, Qualitative Data Analysis. Thousand Oaks, CA: SAGE Publications, 1994.

Publication II

Software Test Automation in Practice: Empirical Observations

Kasurinen, J., Taipale, O. and Smolander, K. (2010), Advances in Software Engineering, Special Issue on Software Test Automation, Hindawi Publishing Co. doi: 10.1155/2010/620836.

Software Test Automation in Practice: Empirical Observations

Jussi Kasurinen, Ossi Taipale, Kari Smolander
Lappeenranta University of Technology, Department of Information Technology, Laboratory of Software Engineering
P.O.Box 20, 53851 Lappeenranta, Finland
jussi.kasurinen@lut.fi, ossi.taipale@lut.fi, kari.smolander@lut.fi

The objective of this industry study was to shed light on the current situation and improvement needs in software test automation. To this end, 55 industry specialists from 31 organizational units were interviewed. In parallel with the survey, a qualitative study was conducted in 12 selected software development organizations. The results indicated that the software testing processes usually follow systematic methods to a large degree, and have only little immediate or critical requirements for resources. Based on the results, the testing processes have approximately three fourths of the resources they need, and have access to a limited, but usually sufficient group of testing tools. As for the test automation, the situation is not as straightforward: based on our study, the applicability of test automation is still limited and its adaptation to testing contains practical difficulties in usability. In this study, we analyze and discuss these limitations and difficulties.

Keywords: software testing, test automation, software industry, case study, survey, grounded theory.

1. INTRODUCTION

Testing is perhaps the most expensive task of a software project. In one estimate, the testing phase took over 50% of the project resources [1]. Besides causing immediate costs, testing is also importantly related to costs related to poor quality, as malfunctioning programs and errors cause large additional expenses to software producers [1, 2]. In one estimate [2], software producers in United States lose annually 21.2 billion dollars because of inadequate testing and errors found by their customers. By adding the expenses caused by errors to software users, the estimate rises to 59.5 billion dollars, of which 22.2 billion could be saved by making investments on testing infrastructure [2]. Therefore improving the quality of software and effectiveness of the testing process can be seen as an effective way to reduce software costs in the long run, both for software developers and users.

One solution for improving the effectiveness of software testing has been applying automation to parts of the testing work. In this approach, testers can focus on critical software features or more complex cases, leaving repetitive tasks to the test automation system. This way it may be possible to use human resources more efficiently, which consequently may contribute to more comprehensive testing or savings in the testing process and overall development budget [3]. As personnel costs and time limitations are significant restraints of the testing processes [4,5], it also seems like a sound investment to develop test automation to get larger coverage with same or even smaller number of testing personnel. Based on market estimates, software companies worldwide invested 931 million dollars in automated software testing tools in 1999, with an estimate of at least 2.6 billion dollars in 2004 [6]. Based on these figures, it seems that the application of test automation is perceived as an important factor of the test process development by the software industry.

The testing work can be divided into manual testing and automated testing. Automation is usually applied to running repetitive tasks such as unit testing or regression testing, where test cases are executed every time changes are made [7]. Typical tasks of test automation systems include development and execution of test scripts and verification of test results. In contrast to manual testing, automated testing is not suitable for tasks in which there is little repetition [8], such as explorative testing or late development verification tests. For these activities manual testing is more suitable, as building automation is an extensive task and feasible only if the case is repeated several times [7,8]. However, the division between automated and manual testing is not as straightforward in practice as it seems; a large concern is also the testability of the software [9], because every piece of code can be made poorly enough to be impossible to test it reliably, therefore making it ineligible for automation.

Software engineering research has two key objectives: the reduction of costs and the improvement of the quality of products [10]. As software testing represents a significant part of quality costs, the successful introduction of test automation infrastructure has a possibility to combine these two objectives, and to overall improve the software testing processes. In a similar prospect, the improvements of the software testing processes are also at the focus point of the new software testing standard ISO 29119 [11]. The objective of the standard is to offer a company-level model for the test processes, offering control, enhancement and follow-up methods for testing organizations to develop and streamline the overall process.

In our prior research project [4, 5, 12, 13, 14], experts from industry and research institutes prioritized issues of software testing using the Delphi method [15]. The experts concluded that process improvement, test automation with testing tools, and the standardization of testing are the most prominent issues in concurrent cost reduction and quality improvement. Furthermore, the focused study on test automation [4] revealed several test automation enablers and disablers which are further elaborated in this study. Our objective is to observe software test automation in practice, and further discuss the applicability, usability and maintainability issues found in our prior research. The general software testing concepts are also observed from the viewpoint of the ISO 29119 model, analysing the test process factors that create the testing strategy in organizations. The approach to achieve these objectives is twofold. First, we wish to explore the software testing practices the organizations are applying and clarify the current status of test automation in the software industry. Secondly, our objective is to identify improvement needs and suggest improvements for the development of software testing and test automation in practice. By understanding these needs, we wish to give both researchers and industry practitioners an insight into tackling the most hindering issues while providing solutions and proposals for software testing and automation improvements.

The study is purely empirical and based on observations from practitioner interviews. The interviewees of this study were selected from companies producing software products and applications at an advanced technical level. The study included three rounds of interviews and a questionnaire, which was filled during the second interview round. We personally visited 31 companies and carried out 55 structured or semi-structured interviews which were tape-recorded for further analysis. The sample selection aimed to represent different polar points of the software industry; the selection criteria were based on concepts such as operating environments, product and application characteristics (e.g. criticality of the products and applications, real time operation), operating domain and customer base.

The paper is structured as follows. First, in Section 2, we introduce comparable surveys and related research. Secondly, the research process and the qualitative and quantitative research methods are described in Section 3. Then the survey results are presented in Section 4 and the interview results in Section 5. Finally, the results and observations and their validity are discussed in Section 6 and closing conclusions in Section 7.

2. RELATED RESEARCH

Besides our prior industry-wide research in testing [4,5,12,13,14], software testing practices and test process improvement have also been studied by others, like Ng *et al.* [16] in Australia. Their study applied the survey method to establish knowledge on such topics as testing methodologies, tools, metrics, standards, training and education. The study indicated that the most common barrier to developing testing was the lack of expertise in adopting new testing methods and the costs associated with testing tools. In their study, only 11 organizations reported that they met testing budget estimates, while 27 organizations spent 1.5 times the estimated cost in testing, and 10 organizations even reported a ratio of 2 or above. In a similar vein, Torkar and Mankefors [17] surveyed different types of communities and organizations. They found that 60% of the developers claimed that verification and validation were the first to be neglected in cases of resource shortages during a project, meaning that even if the testing is important part of the project, it usually is also the first part of the project where cutbacks and downscaling are applied.

As for the industry studies, a similar study approach has previously been used in other areas of software engineering. For example, Ferreira and Cohen [18] completed a technically similar study in South Africa, although their study focused on the application of agile development and stakeholder satisfaction. Similarly, Li *et al.* [19] conducted research on the COTS-based software development process in Norway, and Chen *et al.* [20] studied the application of open source components in software development in China. Overall, case studies covering entire industry sectors are not particularly uncommon [21, 22]. In the context of test automation, there are several studies and reports in test automation practices [such as 23, 24, 25, 26]. However, there seems to be a lack of studies that investigate and compare the practice of software testing automation in different kinds of software development organizations.

In the process of testing software for errors, testing work can be roughly divided into manual and automated testing, which both have individual strengths and weaknesses. For example, Ramler and Wolfmaier [3] summarize the difference between manual and automated testing by suggesting that automation should be used to prevent further errors in working modules, while manual testing is better suited for finding new and unexpected errors. However, how and where the test automation should be used is not so straightforward issue, as the application of test automation seems to be a strongly diversified area of interest. The application of test automation has been studied for example in test case generation [27, 28], GUI testing [29, 30] and workflow simulators [31, 32] to name a few areas. Also according to Bertolino [33], test automation is a significant area of interest in current testing research, with a focus on improving the degree of automation by developing advanced techniques for generating the test inputs, or by finding support procedures such as error report generation to ease the supplemental workload. According to the same study, one of the dreams involving software testing is 100% automated testing. However, for example Bach's [23] study observes that this cannot be achieved, as all automation ultimately requires human intervention, if for nothing else, then at least to diagnose results and maintain automation cases.

The pressure to create resource savings are in many case the main argument for test automation. A simple and straightforward solution for building automation is to apply test automation just on the test cases and tasks that were previously done manually [8]. However, this approach is usually unfeasible. As Persson and Yilmaztürk [26] note, the establishment of automated testing is a costly, high risk project with several real possibilities for failure, commonly called as "pitfalls". One of the most common reasons why creating test automation fails, is that the software is not designed and implemented for testability and reusability, which leads to architecture and tools with low reusability and high maintenance costs. In reality, test automation sets several requisites on a project and has clear enablers and disablers for its suitability [4,24]. In some reported cases [27, 34, 35], it was observed that the application of test automation with an ill-suited process model may be even harmful to the overall process in terms of productivity or cost-effectiveness.

Models for estimating testing automation costs, for example by Ramler and Wolfmaier [3], support decision-making in the trade-off between automated and manual testing. Berner *et al.* [8] also estimate that most of the test cases in one project are run at least five times, and one fourth over 20 times. Therefore the test cases, which are done constantly like smoke tests, component tests and integration tests, seem like ideal place to build test automation. In any case, there seems to be a consensus that test automation is a plausible tool for enhancing quality, and consequently, reducing the software development costs in the long run if used correctly.

Our earlier research on the software test automation [4] has established that test automation is not as straightforward to implement as it may seem. There are several characteristics which enable or disable the applicability of test automation. In this study, our decision was to study a larger group of industry organizations and widen the perspective for further analysis. The objective is to observe, how the companies have implemented test automation and how they have responded to the issues and obstacles that affect its suitability in practice. Another objective is to analyze whether we can identify new kind of hindrances to the application of test automation and based on these findings, offer guidelines on what aspects should be taken into account when implementing test automation in practice.

3. RESEARCH PROCESS

3.1 Research population and selection of the sample

The population of the study consisted of organization units (OUs). The standard ISO/IEC 15504-1 [36] specifies an organizational unit (OU) as a part of an organization that is the subject of an assessment. An organizational unit deploys one or more processes that have a coherent process context and operates within a coherent set of business goals. An organizational unit is typically part of a larger organization, although in a small organization, the organizational unit may be the whole organization.

The reason to use an OU as the unit for observation was that we wanted to normalize the effect of the company size to get comparable data. The initial population and population criteria were decided based on the prior research on the subject. The sample for the first interview round consisted of 12 OUs, which were technically high level units, professionally producing software as their main process. This sample also formed the focus group of our study. Other selection criteria for the sample were based on the polar type selection [37] to cover different types of organizations, for example different businesses, different sizes of the company, and different kinds of operation. Our objective of using this approach was to gain a deep understanding of the cases and to identify, as broadly as possible, the factors and features that have an effect on software testing automation in practice.

For the second round and the survey, the sample was expanded by adding OUs to the study. Selecting the sample was demanding because comparability is not determined by the company or the organization but by comparable processes in the OUs. With the help of national and local authorities (the network of the Finnish Funding Agency for Technology and Innovation) we collected a population of 85 companies. Only one OU from each company was accepted to avoid the bias of over-weighting large companies. Each OU surveyed was selected from a company according to the population criteria. For this round, the sample size was expanded to 31 OUs, which also included the OUs from the first round. The selection for expansion was based on probability sampling; the additional OUs were randomly entered into our database, and every other one was selected for the survey. In the third round, the same sample as in the first round was interviewed. Table 1 introduces the business domains, company sizes and operating areas of our focus OUs. The company size classification is taken from [38].

TABLE 1: Description of the interviewed focus OUs (see also Appendix A).

OU	Business	Company size[1] / Operation
Case A	MES[1] producer and electronics manufacturer	Small / National
Case B	Internet service developer and consultant	Small / National
Case C	Logistics software developer	Large / National
Case D	ICT consultant	Small / National
Case E	Safety and logistics system developer	Medium / National
Case F	Naval software system developer	Medium / International
Case G	Financial software developer	Large / National
Case H	MES[1] producer and logistics service systems provider	Medium / International
Case I	SME[2] business and agriculture ICT service provider	Small / National
Case J	Modeling software developer	Large / International
Case K	ICT developer and consultant	Large / International
Case L	Financial software developer	Large / International

[1] Manufacturing Execution System [2] Small and Medium-sized Enterprise, definition [38]

3.2 Interview rounds

The data collection consisted of three interview rounds. During the first interview round, the designers responsible for the overall software structure and/or module interfaces were interviewed. If the OU did not have separate designers, then the interviewed person was selected from the programmers based on their role in the process to match as closely as possible to the desired responsibilities. The interviewees were also selected so that they came from the same project, or from positions where the interviewees were working on the same product. The interviewees were not specifically told not to discuss the interview questions together, but this behavior was not encouraged either. In a case where an interviewee asked for the questions or interview themes beforehand, the person was allowed access to them in order to prepare for the meeting. The interviews in all three rounds lasted about an hour and had approximately 20 questions related to the test processes or test organizations. In two interviews, there was also more than one person present.

The decision to interview designers in the first round was based on the decision to gain a better understanding on the test automation practice and to see whether our hypothesis based on our prior studies [4, 5, 12, 13, 14] and supplementing literature review were still valid. During the first interview round, we interviewed 12 focus OUs, which were selected to represent different polar types in the software industry. The interviews contained semi-structured questions and were tape-recorded for qualitative analysis. The initial analysis of the first round also provided ingredients for the further elaboration of important concepts for the latter rounds. The interview rounds and the roles of the interviewees in the case OUs are described in Table 2.

TABLE 2: Interviewee roles and interview rounds.

Round type	Number of interviews	Interviewee role	Description
1) Semi-structured	12 focus OUs	Designer or Programmer	The interviewee is responsible for software design or has influence on how software is implemented.
2) Structured/ Semi-structured	31 OUs quantitative, including 12 focus OUs qualitative	Project or testing manager	The interviewee is responsible for software development projects or test processes of software products.
3) Semi-structured	12 focus OUs	Tester	The interviewee is a dedicated software tester or is responsible for testing the software product.

The purpose of the second combined interview and survey round was to collect multiple choice survey data and answers to open questions which were based on the first round interviews. These interviews were also tape-recorded for the qualitative analysis of the focus OUs, although the main data collection method for this round was a structured survey. In this round, project or testing managers from 31 OUs, including the focus OUs, were interviewed. The objective was to collect quantitative data on the software testing process, and further collect material on different testing topics, such as software testing and development. The collected survey data could also be later used to investigate observations made from the interviews and vice versa, as suggested in [38]. Managers were selected for this round, as they tend to have more experience on software projects, and have a better understanding of organizational and corporation level concepts and the overall software process beyond project-level activities.

The interviewees of the third round were testers or, if the OU did not have separate testers, programmers who were responsible for the higher-level testing tasks. The interviews in these rounds were also semi-structured and concerned the work of the interviewees, problems in testing (e.g. increasing complexity of the systems), the use of software components, the influence of the business orientation, testing resources, tools, test automation, outsourcing, and customer influence for the test processes.

The themes in the interview rounds remained similar, but the questions evolved from general concepts to more detailed ones. Before proceeding to the next interview round, all interviews with the focus OUs were transcribed and analyzed because new understanding and ideas emerged during the data analysis. This new understanding was reflected in the next interview rounds. The themes and questions for each of the interview rounds can be found on the project website http://www2.it.lut.fi/project/MASTO/.

3.3 Grounded analysis method

The grounded analysis was used to provide further insight into the software organizations, their software process and testing policies. By interviewing people of different positions from the production organization, the analysis could gain additional information on testing- and test automation-related concepts like different testing phases, test strategies, testing tools and case selection methods. Later this information could be compared between organizations, allowing hypotheses on test automation applicability and the test processes themselves.

The grounded theory method contains three data analysis steps: open coding, axial coding and selective coding. The objective for open coding is to extract the categories from the data, whereas axial coding identifies the connections between the categories. In the third phase, selective coding, the core category is identified and described [39]. In practice, these steps overlap and merge because the theory development process proceeds iteratively. Additionally, Strauss and Corbin [40] state that sometimes the core category is one of the existing categories, and at other times no single category is broad enough to cover the central phenomenon.

TABLE 3: Open coding of the interview data

Interview transcript	Codes, Category: Code
"Well, I would hope for stricter control or management for implementing our testing strategy, as I am not sure if our testing covers everything and is it sophisticated enough. On the other hand, we do have strictly limited resources, so it can be enhanced only to some degree, we cannot test everything. And perhaps, recently we have had, in the newest versions, some regression testing, going through all features, seeing if nothing is broken, but in several occasions this has been left unfinished because time has run out. So there, on that issue we should focus. "	Enhancement proposal: Developing testing strategy Strategy for testing: Ensuring case coverage Problem: Lack of resources Problem: Lack of time

The objective of open coding is to classify the data into categories and identify leads in the data, as shown in the Table 3. The interview data is classified to categories based on the main issue, with observation or phenomenon related to it being the codified part. In general, the process of grouping concepts that seem to pertain to the same phenomena is called categorizing, and it is done to reduce the number of units to work with [40]. In our study, this was done using ATLAS.ti software [41]. The open coding process started with "seed categories" [42] that were formed from the research question and objectives, based on the literature study on software testing and our prior observations [4, 5, 12, 13, 14] on software and testing processes. Some seed categories, like “knowledge management”, “service-orientation” or “approach for software development” were derived from our earlier studies, while categories like “strategy for testing”, “outsourcing”, “customer impact” or “software testing tools” were taken from known issues or literature review observations.

The study followed the approach introduced by Seaman [43], which notes that the initial set of codes (seed categories) comes from the goals of the study, the research questions, and predefined variables of interest. In the open coding, we added new categories and merged existing categories to others if they seemed unfeasible or if we found a better generalization. Especially at the beginning of the analysis, the number of categories and codes quickly accumulated and the total number of codes after open coding amounted to 164 codes in 12 different categories. Besides the test process, software development in general and test automation, these categories also contained codified observations on such aspects as the business orientation, outsourcing, and product quality.

After collecting the individual observations to categories and codes, the categorized codes were linked together based on the relationships observed in the interviews. For example, the codes “Software process: Acquiring 3rd party modules”, “Testing strategy: Testing 3rd party modules”, and “Problem: Knowledge management with 3rd party modules” were clearly related and therefore we connected them together in axial coding. The objective of axial coding is to further develop categories, their properties and dimensions, and find causal, or any kinds of, connections between the categories and codes.

For some categories, the axial coding also makes it possible to define dimension for the phenomenon, for example “Personification-Codification” for “Knowledge management strategy”, where every property could be defined as a point along the continuum defined by the two polar opposites. For the categories that are given dimension, the dimension represented the locations of the property or the attribute of a category [40]. Obviously for some categories, which were used to summarize different observations like enhancement proposals or process problems, defining dimensions was unfeasible. We considered using dimensions for some categories like “criticality of test automation in testing process” or “tool sophistication level for automation tools” in this study, but discarded them later as they yielded only little value to the study. This decision was based on the observation that values of both dimensions were outcomes of the applied test automation strategy, having no effect on the actual suitability or applicability of test automation to the organization’s test process.

Our approach for analysis of the categories included Within-Case Analysis and Cross-Case-Analysis, as specified by Eisenhardt [37]. We used the tactic of selecting dimensions and properties with within-group similarities coupled with inter-group differences [37]. In this strategy, our team isolated one phenomenon that clearly divided the organizations to different groups, and searched for explaining differences and similarities from within these groups. Some of the applied features were, for example, the application of agile development methods, the application of test automation, the size [38] difference of originating companies and service orientation. As for one central result, the appropriateness of OU as a comparison unit was confirmed based on our size difference-related observations on the data; the within-group- and inter-group comparisons did yield results in which the company size or company policies did not have strong influence, whereas the local, within-unit policies did. In addition, the internal activities observed in OUs were similar regardless of the originating company size, meaning that in our study the OU comparison was indeed feasible approach.

We established and confirmed each chain of evidence in this interpretation method by discovering sufficient citations or finding conceptually similar OU activities from the case transcriptions. Finally, in the last phase of the analysis, in selective coding, our objective was to identify the core category [40] – a central phenomenon – and systematically relate it to other categories and generate the hypothesis and the theory. In this study, we consider test automation in practice as the core category, to which all other categories were related as explaining features of applicability or feasibility.

The general rule in grounded theory is to sample until theoretical saturation is reached. This means, until (1) no new or relevant data seem to emerge regarding a category, (2) the category development is dense, insofar as all of the paradigm elements are accounted for, along with variation and process, and (3) the relationships between categories are well established and validated [40]. In our study, the saturation was reached during the third round, where no new categories were created, merged or removed from the coding. Similarly, the attribute values were also stable, i.e. the already discovered phenomena began to repeat themselves in the collected data. As an additional way to ensure the validity of our study and avoid validity threats [44], four researchers took part in the data analysis. The bias caused by researchers was reduced by combining the different views of the researchers (observer triangulation) and a comparison with the phenomena observed in the quantitative data (methodological triangulation) [44,45].

3.4 The survey instrument development and data collection

The survey method described by Fink and Kosecoff [46] was used as the research method in the second round. An objective for a survey is to collect information from people about their feelings and beliefs. Surveys are most appropriate when information should come directly from the people [46]. Kitchenham et al. [47] divide comparable survey studies into exploratory studies from which only weak conclusions can be drawn, and confirmatory studies from which strong conclusions can be drawn. We consider this study as an exploratory, observational, and cross-sectional study that explores the phenomenon of software testing automation in practice and provides more understanding to both researchers and practitioners.

To obtain reliable measurements in the survey, a validated instrument was needed, but such an instrument was not available in the literature. However, Dybå [48] has developed an instrument for measuring the key factors of success in software process improvement. Our study was constructed based on the key factors of this instrument, and supplemented with components introduced in the standards ISO/IEC 29119 [11] and 25010 [49]. We had the possibility to use the current versions of the new standards because one of the authors is a member of the JTC1/SC7/WG26, which is developing the new software testing standard. Based on these experiences a measurement instrument derived from the ISO/IEC 29119 and 25010 standards was used.

The survey consisted of a questionnaire (available at http://www2.it.lut.fi/project/MASTO/) and a face-to-face interview. Selected open-ended questions were located at the end of the questionnaire to cover some aspects related to our qualitative study. The classification of the qualitative answers was planned in advance.

The questionnaire was planned to be answered during the interview to avoid missing answers because they make the data analysis complicated. All the interviews were tape-recorded, and for the focus organizations, further qualitatively analyzed with regard to the additional comments made during the interviews. Baruch [50] also states that the average response rate for self-assisted questionnaires is 55.6%, and when the survey involves top management or organizational representatives the response rate is 36.1%. In this case, a self-assisted, mailed questionnaire would have led to a small sample. For these reasons, it was rejected, and personal interviews were selected instead. The questionnaire was piloted with three OUs and four private persons.

If an OU had more than one respondent in the interview, they all filled the same questionnaire. Arranging the interviews, traveling and interviewing took two months of calendar time. Overall, we were able to accomplish 0.7 survey interviews per working day on an average. One researcher conducted 80% of the interviews, but because of the overlapping schedules also two other researchers participated in the interviews. Out of the contacted 42 OUs, 11 were rejected because they did not fit the population criteria in spite of the source information, or it was impossible to fit the interview into the interviewee's schedule. In a few individual cases, the reason for rejection was that the organization refused to give an interview. All in all, the response rate was, therefore, 74%.

4. TESTING AND TEST AUTOMATION IN SURVEYED ORGANIZATIONS

4.1. General information of the organizational units

The interviewed OUs were parts of large companies (55%) and small and medium-sized enterprises (45%). The OUs belonged to companies developing information systems (11 OUs), IT services (5 OUs), telecommunication (4 OUs), finance (4 OUs), automation systems (3 OUs), the metal industry (2 OUs), the public sector (1 OU), and logistics (1 OU). The application domains of the companies are presented in Figure 1. Software products represented 63% of the turnover, and services (e.g. consulting, subcontracting, and integration) 37%.

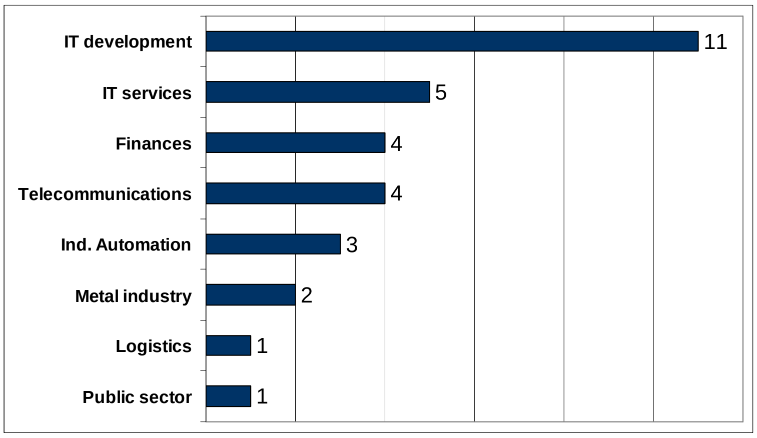


Figure 1. Application domains of the companies.

The maximum number of personnel in the companies to which the OUs belonged was 350 000, the minimum was four, and the median was 315. The median of the software developers and testers in the OUs was 30 persons. OUs applied automated testing less than expected, the median of the automation in testing being 10%. Also, the interviewed OUs utilized agile methods less than expected: the median of the percentage of agile (reactive, iterative) vs. plan driven methods in projects was 30%. The situation with human resources was better than what was expected, as the interviewees estimated that the amount of human resources in testing was 75%. When asking what percent of the development effort was spent on testing, the median of the answers was 25%. The cross-sectional situation of development and testing in the interviewed OUs is illustrated in Table 4.

TABLE 4: Interviewed OUs

	Max.	**Min.**	**Median**
Number of employees in the company.	350 000	4	315
Number of SW developers and testers in the OU.	600	0[1]	30
Percentage of automation in testing.	90	0	10
Percentage of agile (reactive, iterative) vs. plan driven methods in projects.	100	0	30
Percentage of existing testers vs. resources need.	100	10	75
How many percent of the development effort is spent on testing?	70	0[2]	25

[1] 0 means that all of the OUs developers and testers are acquired from 3rd parties
[2] 0 means that no project time is allocated especially for testing

The amount of testing resources was measured by three figures; first the interviewee was asked to evaluate the percentage from total project effort allocated solely to testing. The survey average was 27%, the maximum being 70% and the minimum 0%, meaning that the organization relied solely on testing efforts carried out in parallel with development. The second figure was the amount of test resources compared to the organizational optimum. In this figure, if the company had two testers and required three, it would have translated as 66% of resources. Here the average was 70%; six organizations (19%) reported 100% resource availability. The third figure was the number of automated test cases compared to all of the test cases in all of the test phases the software goes through before its release. The average was 26%, varying between different types of organizations and project types. The results are presented in Figure 2, in which the qualitative study case OUs are also presented for comparison. The detailed descriptions for each case organization are available in Appendix A.

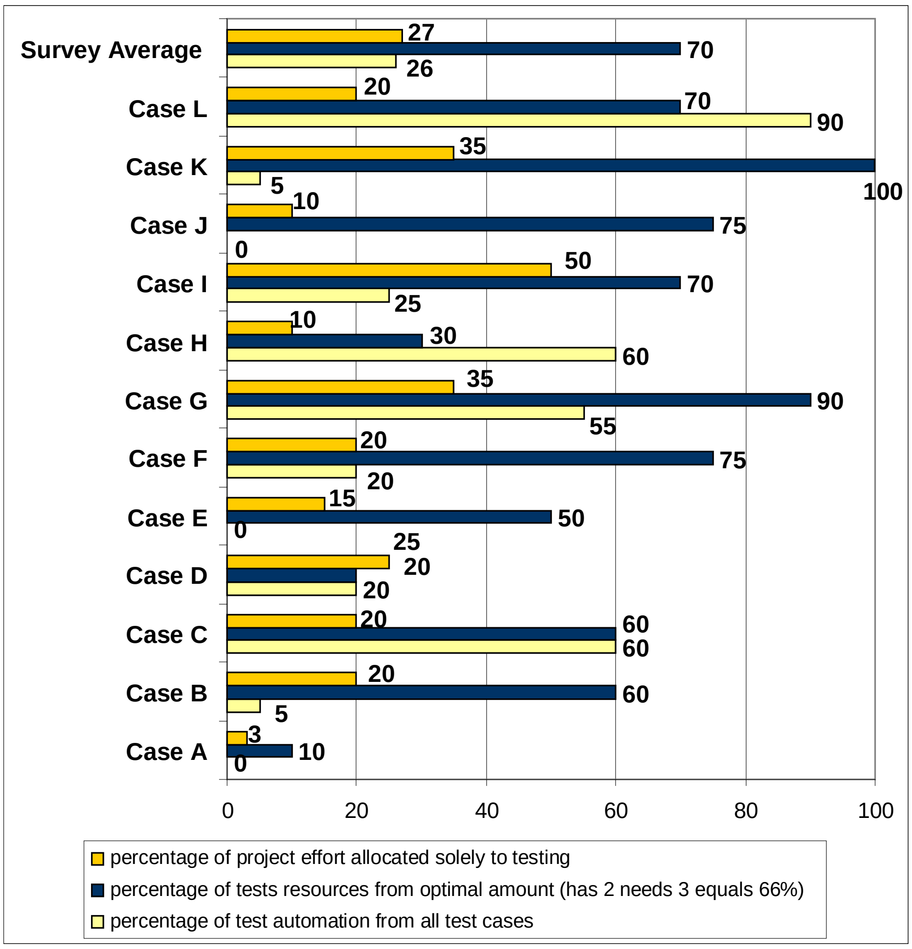


Figure 2. Amount of test resources and test automation in the focus organizations of the study and the survey average.

4.2. General testing items

The survey interviewed 31 organization managers from different types of software industry. The contributions of the interviewees were measured using a five-point Likert scale where 1 denoted “I fully disagree” and 5 denoted “I fully agree”. The interviewees emphasized that quality is built in development (4.3) rather than in testing (2.9). Then the interviewees were asked to estimate their organizational testing practices according to the new testing standard ISO/IEC 29119 [11], which identifies four main levels for testing processes: the test policy, test strategy, test management and testing. The test policy is the company level guideline which defines the management, framework and general guidelines, the test strategy is an adaptive model for the preferred test process, test management is the control level for testing in a software project, and finally, testing is the process of conducting test cases. The results did not make a real difference between the lower levels of testing (test management level and test levels) and higher levels of testing (organizational test policy and organizational test strategy). All in all, the interviewees were rather satisfied with the current organization of testing. The resulted average levels from quantitative survey are presented in Figure 3.

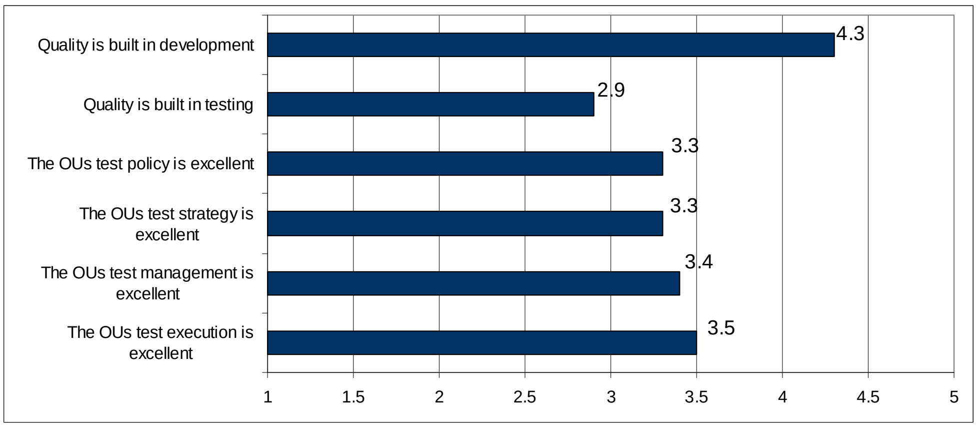


Figure 3. Levels of testing according to the ISO/IEC 29119 standard

Besides the organization, the test processes and test phases were also surveyed. The five-point Likert scale with the same one to five - one being fully disagree and five fully agree - grading method was used to determine the correctness of different testing phases. Overall, the latter test phases - system, functional testing – were considered excellent or very good, whereas the low level test phases such as unit testing and integration received several low-end scores. The organizations were satisfied or indifferent towards all test phases, meaning that there were no strong focus areas for test organization development. However, based on these results it seems plausible that one effective way to enhance testing would be to support low-level testing in unit and integration test phases. The results are depicted in Figure 4.

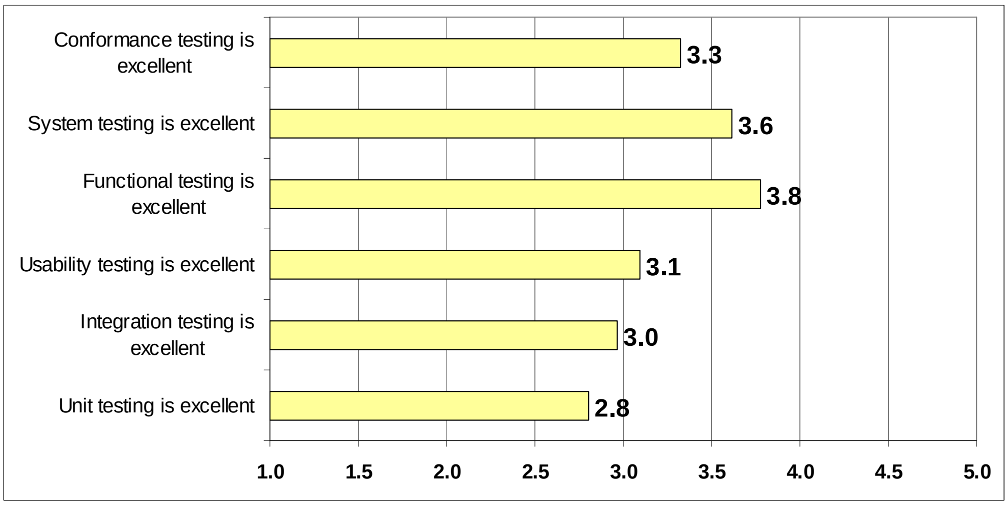


Figure 4. Testing phases in the software process

Finally, the organizations surveyed were asked to rate their testing outcomes and objectives (Figure 5). The first three items discussed the test processes of a typical software project. There seems to be a strong variance in testing schedules and time allocation in the organizations. The outcomes 3.2 for schedule and 3.0 for time allocation do not give any information by themselves, and overall, the direction of answers varied greatly between "Fully disagree" and "Fully agree". However, the situation with test processes was somewhat better; the result 3.5 may also not be a strong indicator by itself, but the answers had only little variance, 20 OUs answering "somewhat agree" or "neutral". This indicates that even if the time is limited and the project schedule restricts testing, the testing generally goes through the normal, defined, procedures.

The fourth and fifth items were related to quality aspects, and gave insights into the clarity of testing objectives. The results of 3.7 for the identification of quality attributes indicate that organizations tend to have objectives for the test processes and apply quality criteria in development. However, the prioritization of their quality attributes is not as strong (3.3) as identification.

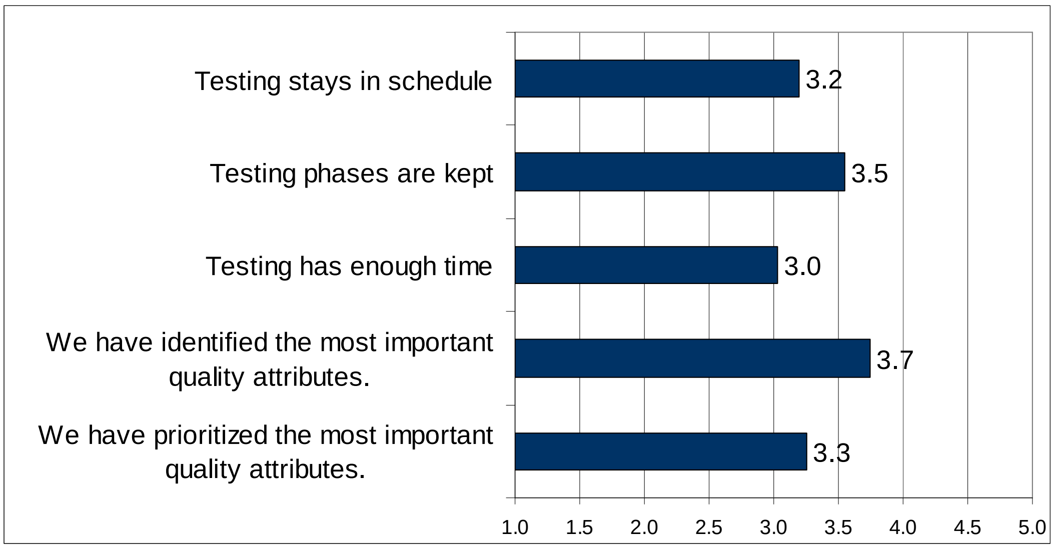


Figure 5. Testing process outcomes

4.3 Testing environment

The quality aspects were also reflected in the employment of systematic methods for the testing work. The majority (61%) of the OUs followed a systematic method or process in the software testing, 13% followed one partially, and 26% of the OUs did not apply any systematic method or process in testing. Process practices were derived from, for example, TPI (Test Process Improvement) [51] or the RUP (Rational Unified Process) [52]. Few Agile development process methods such as Scrum [53] or XP (eXtreme Programming) [54] were also mentioned.

A systematic method is used to steer the software project, but from the viewpoint of testing, the process also needs an infrastructure on which to operate. Therefore, the OUs were asked to report which kind of testing tools they apply to their typical software processes. The test management tools, tools which are used to control and manage test cases and allocate testing resources to cases, turned out to be the most popular category of tools; 15 OUs out of 31 reported the use of this type of tool. The second in popularity were manual unit testing tools (12 OUs), which were used to execute test cases and collect test results. Following them were tools to implement test automation, which were in use in 9 OUs, performance testing tools used in 8 OUs, bug reporting tools in 7 OUs and test design tools in 7 OUs. Test design tools were used to create and design new test cases. The group of other tools consisted of, for example, electronic measurement devices, test report generators, code analyzers, and project management tools. The popularity of the testing tools in different survey organizations is illustrated in Figure 6.

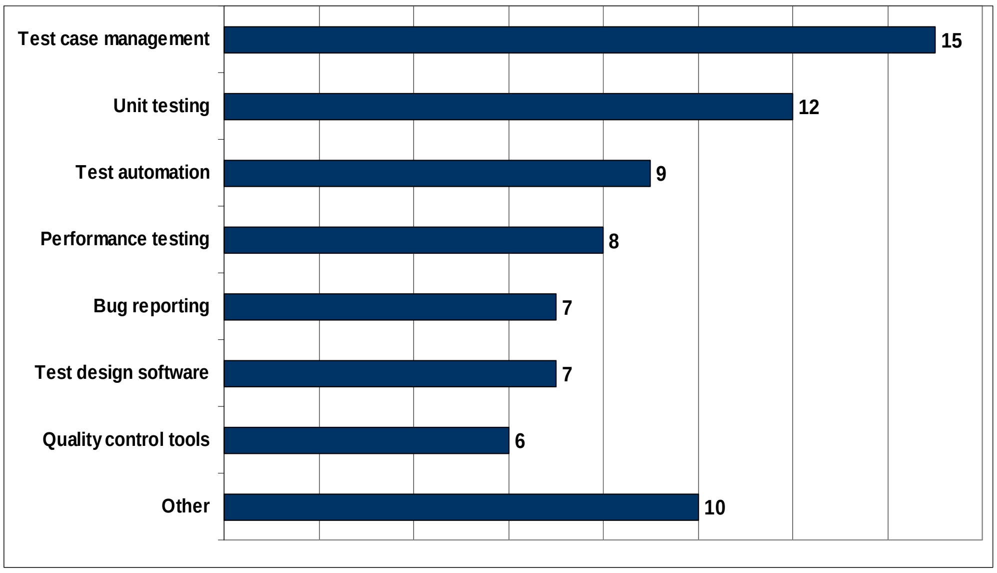


Figure 6. Popularity of the testing tools according to the survey

The respondents were also asked to name and explain the three most efficient application areas of test automation tools. Both the format of the open-ended questions and the classification of the answers were based on the like best (LB) technique adopted from Fink & Kosecoff [46]. According to the LB technique, respondents were asked to list points they considered the most efficient. The primary selection was the area in which the test automation would be the most beneficial to the test organization, the secondary one is the second best area of application, and the third one is the third best area. The interviewees were also allowed to name only one or two areas if they were unable to decide on three application areas. The results revealed the relative importance of software testing tools and methods.

The results are presented in Figure 7. The answers were distributed rather evenly between different categories of tools or methods. The most popular category was unit testing tools or methods (10 interviewees). Next in line were regression testing (9), tools to support testability (9), test environment tools and methods (8) and functional testing (7). The group ‘others’ (11) consisted of conformance testing tools, TTCN-3 (*Testing and Test Control Notation version 3*) tools, general test management tools such as document generators and methods of unit and integration testing. The most popular category, unit testing tools or methods, also received the most primary application area nominations. The most common secondary area of application was regression testing. Several categories ranked third, but concepts such as regression testing, and test environment-related aspects such as document generators were mentioned more than once. Also testability-related concepts - module interface, conformance testing – or functional testing – verification, validation tests – were considered feasible implementation areas for test automation.

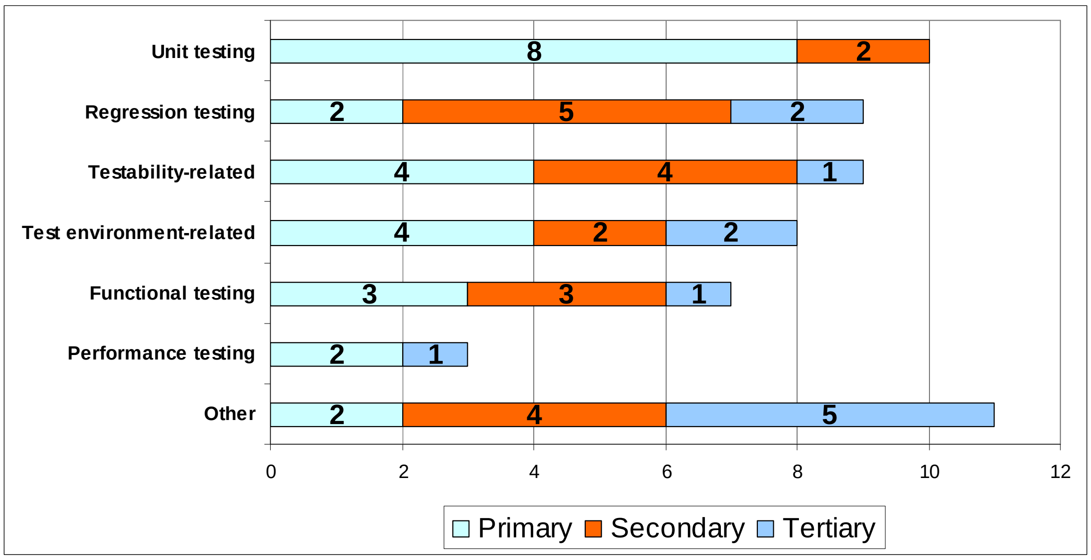


Figure 7. The three most efficient application areas of test automation tools according to the interviewees

4.4 Summary of the survey findings

The survey suggests that interviewees were rather satisfied with their test policy, test strategy, test management, and testing, and did not have any immediate requirements for revising certain test phases, although low-level testing was slightly favoured in the development needs. All in all, 61% of the software companies followed some form of a systematic process or method in testing, with an additional 13% using some established procedures or measurements to follow the process efficiency. The systematic process was also reflected in the general approach to testing; even if the time was limited, the test process followed a certain path, applying the test phases regardless of the project limitations.

The main source of the software quality was considered to be in the development process. In the survey, the test organizations used test automation on an average on 26% of their test cases, which was considerably less than could be expected based on the literature. However, test automation tools were the third most common category of test-related tools, commonly intended to implement unit and regression testing. As for the test automation itself, the interviewees ranked unit testing tools as the most efficient tools of test automation, regression testing being the most common secondary area of application.

5 TEST AUTOMATION INTERVIEWS AND QUALITATIVE STUDY

Besides the survey, the test automation concepts and applications were analyzed based on the interviews with the focus organizations. The grounded theory approach was applied to establish an understanding of the test automation concepts and areas of application for test automation in industrial software engineering. The qualitative approach was applied in three rounds, in which a developer, test manager and tester from 12 different case OUs were interviewed. Descriptions of the case OUs can be found in Appendix A.

In theory-creating inductive research [55], the central idea is that researchers constantly compare theory and data iterating with a theory which closely fits the data. Based on the grounded theory codification, the categories identified were selected in the analysis based on their ability to differentiate the case organizations and their potential to explain the differences regarding the application of test automation in different contexts. We selected the categories so as to explore the types of automation applications and the compatibility of test automation services with the OUs testing organization. We conceptualized the most common test automation concepts based on the coding and further elaborated them to categories to either cater the essential features such as their role in the overall software process or their relation to test automation. We also concentrated on the OU differences in essential concepts such as automation tools, implementation issues or development strategies. This conceptualization resulted to the categories listed in Table 5.

TABLE 5: Test automation categories

Category	Definition
Automation application	Areas of application for test automation in the software process.
Role in software process	The observed roles of test automation in the company software process and the effect of this role.
Test automation strategy	The observed method for selecting the test cases where automation is applied and the level of commitment to the application of test automation in the organizations.
Automation development	The areas of active development in which the OU is introducing test automation.
Automation tools	The general types of test automation tools applied.
Automation issues	The items that hinder test automation development in the OU.

The category "Automation application" describes the areas of software development, where test automation was applied successfully. This category describes the testing activities or phases which apply test automation processes. In the case where the test organization did not apply automation, or had so far only tested it for future applications, this category was left empty. The application areas were generally geared towards regression and stress testing, with few applications of functionality and smoke tests in use.

The category "Role in software process" is related to the objective for which test automation was applied in software development. The role in the software process describes the objective for the existence of the test automation infrastructure; it could, for example, be in quality control, where automation is used to secure module interfaces, or in quality assurance, where the operation of product functionalities is verified. The usual role for the test automation tools was in quality control and assurance, the level of application varying from third party-produced modules to primary quality assurance operations. On two occasions, the role of test automation was considered harmful to the overall testing outcomes, and on one occasion, the test automation was considered trivial, with no real return on investments compared to traditional manual testing.

The category "Test automation strategy" is the approach to how automated testing is applied in the typical software processes, i.e. the way the automation was used as a part of the testing work, and how the test cases and overall test automation strategy were applied in the organization. The level of commitment to applying automation was the main dimension of this category, the lowest level being individual users with sporadic application in the software projects, and the highest being the application of automation to the normal, everyday testing infrastructure, where test automation was used seamlessly with other testing methods and had specifically assigned test cases and organizational support.

The category of "Automation development" is the general category for OU test automation development. This category summarizes the ongoing or recent efforts and resource allocations to the automation infrastructure. The type of new development, introduction strategies and current development towards test automation are summarized in this category. The most frequently chosen code was "general increase of application", where the organization had committed itself to test automation, but had no clear idea of how to develop the automation infrastructure. However, one OU had a development plan for creating a GUI testing environment, while two organizations had just recently scaled down the amount of automation as a result of a pilot project. Two organizations had only recently introduced test automation to their testing infrastructure.

The category of "Automation tools" describes the types of test automation tools that are in everyday use in the OU. These tools are divided based on their technological finesse, varying from self-created drivers and stubs to individual proof-of-concept tools with one specified task to test suites where several integrated components are used together for an effective test automation environment. If the organization had created the tools by themselves, or customized the acquired tools to the point of having new features and functionalities, the category was supplemented with a notification regarding in-house-development.

Finally, the category of "Automation issues" includes the main hindrances which are faced in test automation within the organization. Usually, the given issue was related to either the costs of test automation or the complexity of introducing automation to the software projects which have been initially developed without regards to support for automation. Some organizations also considered the efficiency of test automation to be the main issue, mostly contributing to the fact that two of them had just recently scaled down their automation infrastructure. A complete list of test automation categories and case organizations is given in Table 6.

TABLE 6: Test automation categories affecting the software process in case OUs

Category / OU	Automation application	Role in software process	Test automation strategy	Automation development	Automation tools	Automation issues
Case A	GUI testing, regression testing	Functionality verification	Part of the normal test infrastructure	General increase of application	Individual tools, test suite, in-house development	Complexity of adapting automation to test processes
Case B	Performance, smoke testing	Quality control tool	Part of the normal test infrastructure	GUI testing, unit testing	Individual tools, in-house development	Costs of automation implementation
Case C	Functionality, regression testing, documentation automation	Quality control tool	Part of the normal test infrastructure	General increase of application	Test suite, in-house development	Cost of automation maintenance
Case D	Functionality testing	Quality control for secondary modules	Project-related cases	Upkeep for existing parts	Individual tools	Costs of automation implementation
Case E	System stress testing	Quality assurance tool	Part of the normal test infrastructure	General increase of application	Test suite	Costs of implementing new automation
Case F	Unit and module testing, documentation automation	QC, overall effect harmful	Individual users	Recently scaled down	Individual tools	Manual testing seen more efficient
Case G	Regression testing for use cases	Quality assurance tool	Part of the normal test infrastructure	General increase of application	Test suite	Cost of automation maintenance
Case H	Regression testing for module interfaces	Quality control for secondary modules	Part of the normal test infrastructure	General increase of application	Test suite, in-house development	Underestimation of the effect of automated testing on quality
Case I	Functionality testing	Quality control tool	Project-related cases	Application pilot in development	Proof-of-concept tools	Costs of automation implementation
Case J	Automation not in use	QA, no effect observed	Individual users	Application pilot in development	Proof-of-concept tools	No development incentive
Case K	Small scale system testing	QC, overall effect harmful	Individual users	Recently scaled down	Self-created tools; drivers and stubs	Manual testing seen more efficient
Case L	System stress testing	Verifies module compatibility	Project-related cases	Adapting automation to the testing strategy	Individual tools, in-house development	Complexity of adapting automation to test processes

We elaborated further these properties we observed from the case organizations to create hypotheses for the test automation applicability and availability. These resulting hypotheses were shaped according to advice given by Eisenhardt [37] for qualitative case studies. For example, we perceived the quality aspect as really important for the role of automation in software process. Similarly, the resource needs, especially costs, were much emphasized in the automation issues category. The purpose of the hypotheses below is to summarize and explain the features of test automation that resulted from the comparison of differences and similarities between the organizations.

Hypothesis 1: Test automation should be considered more as a quality control tool rather than a frontline testing method.
The most common area of application observed was functionality verification, i.e. regression testing and GUI event testing. As automation is time-consuming and expensive to create, these were the obvious ways to create test cases which had the minimal number of changes per development cycle. By applying this strategy, organizations could set test automation to confirm functional properties with suitable test cases, and acquire such benefits as support for change management and avoid unforeseen compatibility issues with module interfaces.

"Yes, regression testing, especially automated. It is not manually "hammered in" every time, but used so that the test sets are run, and if there is anything abnormal, it is then investigated." – Manager, Case G

"…had we not used it [automation tests], it would have been suicidal." – Designer, Case D

"It's [automated stress tests] good for showing bad code, how efficient it is and how well designed… stress it enough and we can see if it slows down or even breaks completely." –Tester, Case E

However, there seemed to be some contradicting considerations regarding the applicability of test automation. Cases F, J and K had recently either scaled down their test automation architecture or considered it too expensive or inefficient when compared to manual testing. In some cases, automation was also considered too bothersome to configure for a short-term project, as the system would have required constant upkeep, which was an unnecessary addition to the project workload.

"We really have not been able to identify any major advancements from it [test automation]." – Tester, Case J

"It [test automation] just kept interfering." – Designer, Case K

Both these viewpoints indicated that test automation should not be considered a "frontline" test environment for finding errors, but rather a quality control tool to maintain functionalities. For unique cases or small projects, test automation is too expensive to develop and maintain, and it generally does not support single test cases or explorative testing. However, it seems to be practical in larger projects, where verifying module compatibility or offering legacy support is a major issue.

Hypothesis 2: Maintenance and development costs are common test automation hindrances that universally affect all test organizations regardless of their business domain or company size.
Even though the case organizations were selected to represent different types of organizations, the common theme was that the main obstacles in automation adoption were development expenses and upkeep costs. It seemed to make no difference whether the organization unit belonged to a small or large company, as in the OU levels they shared common obstacles. Even despite the maintenance and development hindrances, automation was considered a feasible tool in many organizations. For example, cases I and L pursued the development of some kind of automation to enhance the testing process. Similarly, cases E and H, which already had a significant number of test automation cases, were actively pursuing a larger role for automated testing.

"Well, it [automation] creates a sense of security and controllability, and one thing that is easily underestimated is its effect on performance and optimization. It requires regression tests to confirm that if something is changed, the whole thing does not break down afterwards." – Designer, Case H

In many cases, the major obstacle for adopting test automation was, in fact, the high requirements for process development resources.

"Shortage of time, resources… we have the technical ability to use test automation, but we don't." – Tester, Case J

"Creating and adopting it, all that it takes to make usable automation… I believe that we don't put any effort into it because it will end up being really expensive." – Designer, Case J

In Case J particularly, the OU saw no incentive in developing test automation as it was considered to offer only little value over manual testing, even if they otherwise had no immediate obstacles other than implementation costs.

Also cases F and K reported similar opinions, as they both had scaled down the amount of automation after the initial pilot projects.

"It was a huge effort to manually confirm why the results were different, so we took it [automation] down." – Tester, Case F

"Well, we had gotten automation tools from our partner, but they were so slow we decided to go on with manual testing." – Tester, Case K

Hypothesis 3: Test automation is applicable to most of the software processes, but requires considerable effort from the organization unit.
The case organizations were selected to represent the polar types of software production operating in different business domains. Out of the focus OUs, there were four software development OUs, five IT service OUs, two OUs from the finance sector and one logistics OU. Of these OUs, only two did not have any test automation, and two others had decided to strategically abandon their test automation infrastructure. Still, the business domains for the remaining organizations which applied test automation were heterogeneously divided, meaning that the business domain is not a strong indicator of whether or not test automation should be applied.

It seems that test automation is applicable as a test tool in any software process, but the amount of resources required for useful automation compared to the overall development resources is what determines whether or not automation should be used. As automation is oriented towards quality control aspects, it may be unfeasible to implement in small development projects where quality control is manageable with manual confirmation. This is plausible, as the amount of required resources does not seem to vary based on aspects beyond the OU characteristics, such as available company resources or testing policies applied. The feasibility of test automation seems to be rather connected to the actual software process objectives, and fundamentally to the decision whether the quality control aspects gained from test automation supersede the manual effort required for similar results.

"… before anything is automated, we should calculate the maintenance effort and estimate whether we will really save time, instead of just automating for automation's sake." –Tester, Case G

"It always takes a huge amount of resources to implement." – Designer, Case A

"Yes, developing that kind of test automation system is almost as huge an effort as building the actual project." – Designer, Case I

Hypothesis 4: The available repertoire of testing automation tools is limited, forcing OUs to develop the tools themselves, which subsequently contributes to the application and maintenance costs.
There were only a few case OUs that mentioned any commercial or publicly available test automation programs or suites. The most common approach to test automation tools was to first acquire some sort of tool for proof-of-concept piloting, then develop similar tools as in-house-production or extend the functionalities beyond the original tool with the OU's own resources. These resources for in-house-development and upkeep for self-made products are one of the components that contribute to the costs of applying and maintaining test automation.

"Yes, yes. That sort of [automation] tools have been used, and then there's a lot of work that we do ourselves. For example, this stress test tool…" – Designer, Case E

"We have this 3rd party library for the automation. Well, actually, we have created our own architecture on top of it…" – Designer, Case H

"Well, in [company name], we've-, we developed our own framework to, to try and get around some of these, picking which tests, which group of tests should be automated." – Designer, Case C

However, it should be noted that even if the automation tools were well-suited for the automation tasks, the maintenance still required significant resources if the software product to which it was connected was developing rapidly.

"Well, there is the problem [with automation tool] that sometimes the upkeep takes an incredibly large amount of time." – Tester, Case G

"Our system keeps constantly evolving, so you'd have to be constantly recording [maintaining tools]…" – Tester, Case K

6. DISCUSSION

An exploratory survey combined with interviews was used as the research method. The objective of this study was to shed light on the status of test automation and to identify improvement needs in and the practice of test automation. The survey revealed that the total effort spent on testing (median 25%) was less than expected. The median percentage (25%) of testing is smaller than the 50-60% that is often mentioned in the literature [38, 39]. The comparable low percentage may indicate that that the resources needed for software testing are still underestimated even though testing efficiency has grown. The survey also indicated that companies used fewer resources on test automation than expected: on an average 26% of all of the test cases apply automation. However, there seems to be ambiguity as to which activities organizations consider test automation, and how automation should be applied in the test organizations. In the survey, several organizations reported that they have an extensive test automation infrastructure, but this did not reflect on the practical level, as in the interviews with testers particularly, the figures were considerably different. This indicates that the test automation does not have strong strategy in the organization, and has yet to reach maturity in several test organizations. Such concepts as quality assurance testing and stress testing seem to be particularly unambiguous application areas, as the cases E and L demonstrated. In Case E, the management did not consider stress testing an automation application, whereas testers did. Moreover, in Case L the large automation infrastructure did not reflect on the individual project level, meaning that the automation strategy may strongly vary between different projects and products even within one organization unit.

The qualitative study which was based on interviews indicated that some organizations, in fact, actively avoid using test automation, as it is considered to be expensive and to offer only little value for the investment. However, test automation seems to be generally applicable to the software process, but for small projects the investment is obviously oversized. One additional aspect that increases the investment are tools, which unlike in other areas of software testing, tend to be developed in-house or are heavily modified to suit specific automation needs. This development went beyond the localization process which every new software tool requires, extending even to the development of new features and operating frameworks. In this context it also seems plausible that test automation can be created for several different test activities. Regression testing, GUI testing or unit testing, activities which in some form exist in most development projects, all make it possible to create successful automation by creating suitable tools for the task, as in each phase can be found elements that have sufficient stability or unchangeability. Therefore it seems that the decision on applying automation is not only connected to the enablers and disablers of test automation [4], but rather on tradeoff of required effort and acquired benefits; In small projects or with low amount of reuse the effort becomes too much for such investment as applying automation to be feasible.

The investment size and requirements of the effort can also be observed on two other occasions. First, test automation should not be considered as an active testing tool for finding errors, but as a quality control tool to guarantee the functionality of already existing systems. This observation is in line with those of Ramler and Wolfmaier [3], who discuss the necessity of a large number of repetitive tasks for the automation to supersede manual testing in cost-effectiveness, and of Berner *et al.* [8], who notify that the automation requires a sound application plan and well-documented, simulatable and testable objects. For both of these requirements, quality control at module interfaces and quality assurance on system operability are ideal, and as it seems, they are the most commonly used application areas for test automation. In fact, Kaner [56] states that 60-80% of the errors found with test automation are found in the development phase for the test cases, further supporting the quality control aspect over error discovery.

Other phenomena that increase the investment are the limited availability and applicability of automation tools. On several occasions, the development of the automation tools was an additional task for the automation-building organization that required the organization to allocate their limited resources to the test automation tool implementation. From this viewpoint it is easy to understand why some case organizations thought that manual testing is sufficient and even more efficient when measured in resource allocation per test case. Another approach which could explain the observed resistance to applying or using test automation was also discussed in detail by Berner *et al.* [8], who stated that organizations tend to have inappropriate strategies and overly ambitious objectives for test automation development, leading to results that do not live up to their expectations, causing the introduction of automation to fail. Based on the observations regarding the development plans beyond piloting, it can also be argued that the lack of objectives and strategy also affect the successful introduction processes.

Similar observations of “automation pitfalls” were also discussed by Persson and Yilmaztürk [26] and Mosley and Posey [57].

Overall, it seems that the main disadvantages of testing automation are the costs, which include implementation costs, maintenance costs, and training costs. Implementation costs included direct investment costs, time, and human resources. The correlation between these test automation costs and the effectiveness of the infrastructure are discussed by Fewster [24]. If the maintenance of testing automation is ignored, updating an entire automated test suite can cost as much, or even more than the cost of performing all the tests manually, making automation a bad investment for the organization. We observed this phenomenon in two case organizations. There is also a connection between implementation costs and maintenance costs [24]. If the testing automation system is designed with the minimization of maintenance costs in mind, the implementation costs increase, and vice versa. We noticed the phenomenon of costs preventing test automation development in six cases. The implementation of test automation seems to be possible to accomplish with two different approaches: by promoting either maintainability or easy implementation. If the selected focus is on maintainability, test automation is expensive, but if the approach promotes easy implementation, the process of adopting testing automation has a larger possibility for failure. This may well be due to the higher expectations and assumption that the automation could yield results faster when promoting implementation over maintainability, often leading to one of the automation pitfalls [26] or at least a low percentage of reusable automation components with high maintenance costs.

7. CONCLUSIONS

The objective of this study was to observe and identify factors that affect the state of testing, with automation as the central aspect, in different types of organizations. Our study included a survey in 31 organizations and a qualitative study in 12 focus organizations. We interviewed employees from different organizational positions in each of the cases.

This study included follow-up research on prior observations [4, 5, 12, 13, 14] on testing process difficulties and enhancement proposals, and on our observations on industrial test automation [4]. In this study we further elaborated on the test automation phenomena with a larger sample of polar type OUs, and more focused approach on acquiring knowledge on test process-related subjects. The survey revealed that test organizations use test automation only in 26% of their test cases, which was considerably less than could be expected based on the literature. However, test automation tools were the third most common category of test-related tools, commonly intended to implement unit and regression testing. The results indicate that adopting test automation in software organization is a demanding effort. The lack of existing software repertoire, unclear objectives for overall development and demands for resource allocation both for design and upkeep create a large threshold to overcome.

Test automation was most commonly used for quality control and quality assurance. In fact, test automation was observed to be better suited to such tasks than to actual front-line testing, where the purpose is to find as many faults as possible. However, the high implementation and maintenance requirements were considered the most important issues hindering test automation development, limiting the application of test automation in most OUs. Furthermore, the limited availability of test automation tools and the level of commitment required to develop a suitable automation infrastructure caused additional expenses. Due to the high maintenance requirements and low return on investments in small-scale application, some organizations had actually discarded their automation systems or decided not to implement test automation. The lack of a common strategy for applying automation was also evident in many interviewed OUs. Automation applications varied even within the organization, as was observable in the differences when comparing results from different stakeholders. In addition, the development strategies were vague and lacked actual objectives. These observations can also indicate communication gaps [58] between stakeholders of the overall testing strategy, especially between developers and testers.

The data also suggested that the OUs that had successfully implemented test automation infrastructure to cover the entire organization seemed to have difficulties in creating a continuance plan for their test automation development. After the adoption phases were over, there was an ambiguity about how to continue, even if the organization had decided to further develop their test automation infrastructure. The overall objectives were usually clear and obvious – cost savings and better test coverage – but in practise there were only few actual development ideas and novel concepts. In the case organizations this was observed in the vagueness of the development plans: only one of the five OUs which used automation as a part of their normal test processes had development plans beyond the general will to increase the application.

The survey established that 61% of the software companies followed some form of a systematic process or method in testing, with an additional 13% using some established procedures or measurements to follow the process efficiency. The main source of software quality was considered to reside in the development process, with testing having much smaller impact in the product outcome. In retrospect of the test levels introduced in the ISO/IEC29119 standard, there seems to be no one particular level of the testing which should be the research and development interest for best result enhancements. However, the results from the self-assessment of the test phases indicate that low-level testing could have more potential for testing process development.

Based on these notions, the research and development should focus on uniform test process enhancements, such as applying a new testing approach and creating an organization-wide strategy for test automation. Another focus area should be the development of better tools to support test organizations and test processes in the low-level test phases such as unit or integration testing. As for automation, one tool project could be the development of a customizable test environment with a common core and with an objective to introduce less resource-intensive, transferable and customizable test cases for regression and module testing.

8. ACKNOWLEDGEMENTS

This study is a part of the ESPA project (http://www.soberit.hut.fi/espa/), funded by the Finnish Funding Agency for Technology and Innovation (project number 40125/08) and by the participating companies listed on the project web site.

REFERENCES

[1] Kit, E., Software Testing in the Real World: Improving the Process. Addison-Wesley, Reading, MA, USA, 1995.

[2] Tassey, G., The Economic Impacts of Inadequate Infrastructure for Software Testing. U.S. National Institute of Standards and Technology report, RTI Project Number 7007.011, 2002.

[3] Ramler, R. and Wolfmaier, K. Observations and lessons learned from automated testing. Proceedings of the 2006 international workshop on Automation of software testing, Shanghai, China, Pages: 85 – 91, 2006.

[4] Karhu, K., Repo, T., Taipale, O. and Smolander, K., Empirical Observations on Software Testing Automation, Proceeding of the 2nd International Conference on Software Testing, Verification and Validation, Denver, CO, USA, 2009.

[5] Taipale, O. and Smolander, K. Improving Software Testing by Observing Causes, Effects, and Associations from Practice. the International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, 2006.

[6] Shea, B., Sofware Testing Gets New Respect, InformationWeek, July 3 issue, 2000.

[7] Dustin, E., Rashka, J. and Paul, J., Automated software testing: introduction, management, and performance. Addison-Wesley, Boston, 1999.

[8] Berner, S., Weber, R. and Keller, R.K. Observations and lessons learned from automated testing. Proceedings of the 27th international conference on Software engineering, St. Louis, MO, USA, Pages: 571 – 579, 2005

[9] Whittager, J.A., What is Software Testing? And Why is it So Hard?, IEEE Software, 17(1), pages 70-79, 2000.

[10] Osterweil, L.J., Software processes are software too, revisited: an invited talk on the most influential paper of ICSE 9, presented at the International Conference on Software Engineering, Proc. 19th International Conference on Software Engineering, Boston, 1997.

[11] ISO/IEC, ISO/IEC 29119-2, Software Testing Standard – Activity Descriptions for Test Process Diagram, 2008.

[12] Taipale, O., Smolander, K. and Kälviäinen, H. Cost Reduction and Quality Improvement in Software Testing. Software Quality Management Conference, Southampton, UK, 2006.

[13] Taipale, O., Smolander, K. and Kälviäinen, H. Factors Affecting Software Testing Time Schedule. the Australian Software Engineering Conference, Sydney. IEEE Comput. Soc, Los Alamitos, CA, USA, 2006.

[14] Taipale, O., Smolander, K. and Kälviäinen, H. A Survey on Software Testing. 6th International SPICE Conference on Software Process Improvement and Capability dEtermination (SPICE'2006), Luxembourg, 2006.

[15] Dalkey, N.C., The Delphi method: An experimental study of group opinion, RAND Corporation, Santa Monica, CA 1969.

[16] Ng, S.P., Murmane, T., Reed, K., Grant, D. and Chen, T.Y., A Preliminary Survey on Software Testing Practices in Australia, in Proc. 2004 Australian Software Engineering Conference (Melbourne, Australia), Pages 116-125, 2004.

[17] Torkar, R. and Mankefors, S., A survey on testing and reuse, presented at the IEEE International Conference on Software - Science, Technology and Engineering (SwSTE'03), Herzlia, Israel, 2003

[18] Ferreira, C. & Cohen, J. Agile Systems Development and Stakeholder Satisfaction: A South African Empirical Study, Proc. SAICSIT 2008, Wilderness, South Africa, Pages 48-55, 2008.

[19] Li, J., Bjørnson, F.O., Conradi R. and Kampenes, V.B. An empirical study of variations in COTS-based software development processes in the Norwegian IT industry, Empirical Software Engineering, 11(3), 2006.

[20] Chen, W., Li, J., Ma, J., Conradi, R., Ji, J. and Liu, C. An empirical study on software development with open source components in the Chinese software industry, Software Process: Improvement and Practice, 13(1), 2008.

[21] Dossani R. and Denny, N. The Internet's role in offshored services: A case study of India, ACM Transactions on Internet Technology (TOIT), 7(3), 2007.

[22] Wong, K.Y. An exploratory study on knowledge management adoption in the Malaysian industry, International Journal of Business Information Systems, 3(3), 2008.

[23] Bach, J. Test Automation Snake Oil, Proc. 14th International Conference and Exposition on Testing Computer Software, 1999.

[24] Fewster, M. Common Mistakes in Test Automation, Grove Consultants, 2001.

[25] Hartman, A., Katara, M. and Paradkar, A., Domain specific approaches to software test automation. Proc. 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, Dubrovnik, Croatia, Pages: 621-622, 2007.

[26] Persson, C. and Yilmazturk, N. Establishment of automated regression testing at ABB: industrial experience report on 'avoiding the pitfalls'. Proceedings of the 19th International Conference on Automated Software Engineering, Pages: 112- 121, 2004.

[27] Auguston, M., Michael, J.B. and Shing, M.-T. Test automation and safety assessment in rapid systems prototyping. The 16th IEEE International Workshop on Rapid System Prototyping, Montreal, Canada, Pages: 188-194, 2005.

[28] Cavarra, A., Davies, J., Jeron, T., Mournier, L., Hartman, A. and Olvovsky, S., Using UML for Automatic Test Generation, Proceedings of ISSTA'2002, Aug. 2002.

[29] Vieira, M., Leduc, J., Subramanyan, R. and Kazmeier, J. Automation of GUI testing using a model-driven approach. Proceedings of the 2006 international workshop on Automation of software testing, Shanghai, China, Pages: 9 – 14, 2006.

[30] Xiaochun, Z., Bo, Z., Juefeng, L. and Qiu, G., A test automation solution on GUI functional test, 6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008, 13-16 July, Pages: 1413 – 1418, 2008.

[31] Kreuer, D., Applying test automation to type acceptance testing of telecom networks: a case study with customer participation, 14th IEEE International Conference on Automated Software Engineering, 12-15 Oct., Pages: 216-223, 1999.

[32] Yu, W.D. and Patil, G., A Workflow-Based Test Automation Framework for Web Based Systems, 12th IEEE Symposium on Computers and Communications, 2007. ISCC 2007,1-4 July, Pages: 333 – 339, 2007.

[33] Bertolino, A. Software Testing Research: Achievements, Challenges, Dreams, in Future of Software Engineering: IEEE Computer Society, Pages: 85-103, 2007.

[34] Blackburn, M., Busser, R. and Nauman, A. Why Model-Based Test Automation is Different and What You Should Know to Get Started. International Conference on Practical Software Quality, 2004.

[35] Santos-Neto, P., Resende, R. and Pádua, C. Requirements for information systems model-based testing, Proceedings of the 2007 ACM symposium on Applied computing, Seoul, Korea, Pages: 1409 – 1415, 2007.

[36] ISO/IEC, ISO/IEC 15504-1, Information Technology - Process Assessment - Part 1: Concepts and Vocabulary, 2002.

[37] Eisenhardt, K. M., Building Theories from Case Study Research. Academy of Management Review. 14, Pages: 532-550, 1989.

[38] EU, European Commission, The New SME definition: User guide and model declaration, 2003.

[39] Paré, G. and Elam, J. J. Using Case Study Research to Build Theories of IT Implementation. The IFIP TC8 WG International Conference on Information Systems and Qualitative Research, Philadelphia, USA. Chapman & Hall, 1997.

[40] Strauss, A. and Corbin, J., Basics of Qualitative Research: Grounded Theory Procedures and Techniques. SAGE Publications, Newbury Park, CA, USA, 1990.

[41] ATLAS.ti - The Knowledge Workbench. Scientific Software Development, 2005.

[42] Miles, M. B. and Huberman, A. M. Qualitative Data Analysis. SAGE Publications, Thousand Oaks, CA, USA, 1994.

[43] Seaman, C. B., Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering. 25, Pages: 557-572, 1999.

[44] Robson, C., Real World Research, Second Edition. Blackwell Publishing, 2002.

[45] Denzin, N. K., The research act: A theoretical introduction to sociological methods. McGraw-Hill,1978.

[46] Fink, A. and Kosecoff, J. *How to Conduct Surveys: A Step-By-Step Guide.* Beverly Hills, CA: SAGE, 1985.

[47] Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., Emam, K.E. and Rosenberg, J., Preliminary Guidelines for Empirical Research in Software Engineering, IEEE Transactions on Software Engineering, 28, No. 8, Pages: 721-733, 2002.

[48] Dybå, T., An Instrument for Measuring the Key Factors of Success in Software Process Improvement, Empirical Software Engineering, 5, Pages: 357-390, 2000.

[49] ISO/IEC, ISO/IEC 25010-2, Softaware Engineering – Software product Quality Requirements and Evaluation (SQuaRE) Quality Model, 2008.

[50] Baruch, Y., Response Rate in Academic Studies - A Comparative Analysis, Human Relations, 52(4), Pages: 421-438, 1999.

[51] Koomen, T. and Pol, M., Test Process Improvement: a Practical Step-by-Step Guide to Structured Testing, Addison-Wesley, 1999.

[52] Kruchten, P., The Rational Unified Process: An Introduction, second edition. Addison-Wesley Professional, 1998.

[53] Schwaber, K. and Beedle, M., Agile software development with Scrum, Prentice Hall, 2001.

[54] Beck, K., Extreme Programming Explained: Embrace Change, 2000.

[55] Glaser, B. and Strauss, A. L., The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine, Chicago, 1967.

[56] Kaner, C. Improving the Maintainability of Automated Test Suites. Software QA, 4(4), 1997.

[57] Mosley D.J. and Posey B.A. *Just Enough Software Test Automation*. Prentice Hall, 2002.

[58] Foray, D., Economics of Knowledge. The MIT Press, Cambridge, MA, 2004.

APPENDIX A: CASE DESCRIPTIONS

Case A, Manufacturing execution system (MES) producer and electronics manufacturer. Case A produces software as a service (SaaS) for their product. The company is a small-sized, nationally operating company that has mainly industrial customers. Their software process is a plan-driven cyclic process, where the testing is embedded to the development itself, having only little amount of dedicated resources. This organization unit applied test automation as a user interface and regression testing tool, using it for product quality control. Test automation was seen as a part of the normal test strategy, universally used in all software projects. The development plan for automation was to generally increase the application, although the complexity of the software- and module architecture was considered major obstacle on the automation process.

Case B, Internet service developer and consultant. Case B organization offers two types of services; development of Internet service portals for the customers like communities and public sector, and consultation in the Internet service business domain. The origination company is small and operates on a national level. Their main resource on the test automation is in the performance testing as a quality control tool, although addition of GUI test automation has also been proposed. The automated tests are part of the normal test process, and the overall development plan was to increase the automation levels especially to the GUI test cases. However, this development has been hindered by the cost of designing and developing test automation architecture.

Case C, Logistics software developer. Case C organization focuses on creating software and services for their origin company and its customers. This organization unit is a part of a large-sized, nationally operating company with large, highly distributed network and several clients. The test automation is widely used in several testing phases like functionality testing, regression testing and document generation automation. These investments are used for quality control to ensure the software usability and correctness. Although the OU is still aiming for larger test automation infrastructure, the large amount of related systems and constant changes within the inter-module communications is causing difficulties in development and maintenance of the new automation cases.

Case D, ICT consultant. Case D organization is a small, regional software consultant company, whose customers mainly compose of small business companies and the public sector. Their organization does some software development projects, in which the company develops services and ICT products for their customers. The test automation comes mainly trough this channel, as the test automation is mainly used as a conformation test tool for the third party modules. This also restricts the amount of test automation to the projects, in which these modules are used. The company currently does not have development plans for the test automation as it is considered unfeasible investment for the OU this size, but they do invest on the upkeep of the existing tools as they have usage as a quality control tool for the acquired outsider modules.

Case E, Safety and logistics system developer. Case E organization is a software system developer for safety and logistics systems. Their products have high amount of safety critical features and have several interfaces on which to communicate with. The test automation is used as a major quality assurance component, as the service stress tests are automated to a large degree. Therefore the test automation is also a central part of the testing strategy, and each project has defined set of automation cases. The organization is aiming to increase the amount of test automation and simultaneously develop new test cases and automation applications for the testing process. The main obstacle for this development has so far been the costs of creating new automation tools and extending the existing automation application areas.

Case F, Naval software system developer. The Case F organization unit is responsible for developing and testing naval service software systems. Their product is based on a common core, and has considerable requirements for compatibility with the legacy systems. This OU has tried test automation on several cases with application areas such as unit- and module testing, but has recently scaled down test automation for only support aspects such as the documentation automation. This decision was based on the resource requirements for developing and especially maintaining the automation system, and because the manual testing was in this context considered much more efficient as there were too much ambiguity in the automation-based test results.

Case G, Financial software developer. Case G is a part of a large financial organization, which operates nationally but has several internationally connected services due to their business domain. Their software projects are always aimed as a service portal for their own products, and have to pass considerable verification and validation tests before being introduced to the public. Because of this, the case organization has sizable test department when compared to other case companies in this study, and follows rigorous test process plan in all of their projects. The test automation is used in the regression tests as a quality assurance tool for user interfaces and interface events, and therefore embedded to the testing strategy as a normal testing environment. The development plans for the test automation is aimed to generally increase the amount of test cases, but even the existing test automation infrastructure is considered expensive to upkeep and maintain.

Case H, Manufacturing execution system (MES) producer and logistics service system provider. Case H organization is a medium-sized company, whose software development is a component for the company product. Case organization products are used in logistics service systems, usually working as a part of automated processes. The case organization applies automated testing as a module interface testing tool, applying it as a quality control tool in the test strategy. The test automation infrastructure relies on the in-house-developed testing suite, which enables organization to use the test automation to run daily tests to validate module conformance. Their approach on the test automation has been seen as a positive enabler, and the general trend is towards increasing automation cases. The main test automation disability is considered to be that the quality control aspect is not visible when working correctly and therefore the effect of test automation may be underestimated in the wider organization.

Case I, Small and medium-sized enterprise (SME) business and agriculture ICT-service provider. The case I organization is a small, nationally operating software company which operates on multiple business domain. Their customer base is heterogeneous, varying from finances to the agriculture and government services. The company is currently not utilizing test automation in their test process, but they have development plans for designing quality control automation. For this development they have had some individual proof-of-concept tools, but currently the overall testing resources limit the application process.

Case J, Modeling software developer. Case J organization develops software products for civil engineering and architectural design. Their software process is largely plan-driven with rigorous verification and validation processes in the latter parts of an individual project. Even though the case organization itself has not implemented test automation, on the corporate level there are some pilot projects where regression tests have been automated. These proof-of-concept-tools have been introduced to the case OU and there are intentions to apply them in the future, but there has so far been no incentive for adoption of the automation tools, delaying the application process.

Case K, ICT developer and consultant. Case K organization is a large, international software company which offers software products for several business domains and government services. Case organization has previously piloted test automation, but decided against adopting the system as it was considered too expensive and resource-intensive to maintain when compared to the manual testing. However, some of these tools still exist, used by individual developers along with test drivers and interface studs in unit- and regression testing.

Case L, Financial software developer. Case L organization is a large software provider for their corporate customer which operates on the finance sector. Their current approach on software process is plan-driven, although some automation features has been tested on a few secondary processes. The case organization does not apply test automation as is, although some module stress test cases have been automated as pilot tests. The development plan for test automation is to generally implement test automation as a part of their testing strategy, although amount of variability and interaction in the module interfaces is considered difficult to implement in test automation cases.

Publication III

A Study of Agility and Testing Processes in Software Organizations

Kettunen, V., Kasurinen, J., Taipale, O. and Smolander, K. (2010), Proceedings of the 19th international symposium on Software testing and analysis (ISSTA), 12.-16.7.2010, Trento, Italy, doi: 10.1145/1831708.1831737

A Study on Agility and Testing Processes in Software Organizations

Vesa Kettunen, Jussi Kasurinen, Ossi Taipale, and Kari Smolander
Lappeenranta University of Technology, Laboratory of Software Engineering
Skinnarilankatu 34, P.O. Box 20
FI-53851 Lappeenranta, Finland
+358 400 213 864

vesa.kettunen | jussi.kasurinen | ossi.taipale | kari.smolander@lut.fi

ABSTRACT

In this paper, we studied the differences in testing activities between software organizations which apply agile development methods and organizations which take the traditional plan-driven approach. Our focus was on the concepts which allow the software organization to successfully apply agile development methods or plan-driven methods. We also observed the test process enhancements and hindrances, which originate in the selected development method. We interviewed 12 organizations, which were selected to represent different polar types of software production. The interviews were tape-recorded and transcribed for further analysis. The study yielded hypotheses which were derived by applying the qualitative grounded theory method. The results indicated that in practice, agile methods can improve the position of testing through the early involvement of testing activities in development, and also have a positive influence on end-product satisfaction. By applying these results, organizations can improve their processes and avoid pitfalls when transitioning to agile methods.

Categories and Subject Descriptors

D.2.9 [****Software Engineering**]: Management – *Productivity*, D.2.8 [****Software Engineering**]: Metrics – *Process metrics*

General Terms

Management, Economics, Human Factors

Keywords

Agile development, test process, case study, empirical study.

1. INTRODUCTION

Several different approaches can be applied to software development. The software development process is based on traditional plan-driven methods such as prototyping [7] or waterfall [27] or agile development methods such as SCRUM [24] or Extreme Programming (XP) [2]. In theory, the main incentive for applying agile development in industry is to gain cost savings and faster development cycles [11], as the development is focused on communication and reacting to the process objectives instead of excessive design. Both plan-driven and agile development methods have their advantages and disadvantages. Hence it is seen that they have their own home grounds [4] and project types they fit into. However, it can be argued [8, 19] that agile development may require some adjustments in the developer organization. For example, one difficulty in the application of agile methods could be fitting the testing activities into the production process while ensuring that the scope of testing is sufficient and designing the test cases around a rapidly evolving software product. As testing is one of the most costly parts of the software process [3] – a common estimation is at least 50 percent of the development costs - testing activities in agile development are an interesting field of research.

Applying agile development methods has effects on the software test process. Several agile methods promote early testing activities as a part of the program, meaning that the test phase is involved in the development. In that way testing, is not left as the final phase of development, as in plan-driven methods, and there may be more time to perform testing activities and correct defects found in testing. However, most of the agile methods do not give enough guidance on how testing should be arranged in parallel with development work [12]. Testing carried out by developers is emphasized, and therefore, the issue is raised whether testers are needed in agile development at all [31].

In this study, we observed the effect of agile development methods from the viewpoint of the test process and test activities. Our goal was to determine how agile methods affect the execution of software testing in practice. The focus was on the organization of testing and on the test strategy. The study was conducted as a qualitative one using the grounded theory research method [29]. The research material consisted of interview data collected from three different interviewees from each of the twelve organization units participating in the study.

This paper is a part of a larger study conducted by our research group on software organizations and test processes. Our earlier publications have covered such topics as test automation [15], test process difficulties and enhancement proposals [17], and test resources, testing tools and testing practices [16].

The paper is structured as follows: Section 2 presents related research from relevant fields, and Section 3 introduces the research method and data collection methods applied in this study. Section 4 presents the results and the hypothesis. Section 5 discusses the findings and the observations, and Section 6 closes the paper with conclusions.

2. RELATED RESEARCH

Earlier studies on software testing with agile methods have been mostly case studies. For example, several studies [22,26,28,30,31] have made experimental observations in one organization when applying agile methods. The authors study the topic by concentrating on their experiences of testing practices in an agile development environment. However, a wider perspective is taken in a study by Itkonen et al. [12]. In addition, Ambler [1] takes software testing into account in his study that considers scalability of agile software development. Ambler introduces a team of independent testers in the application of agile methods at scale.

In the study by Talby et al. [31], testing carried out by developers is emphasized because quality assurance is every team member's responsibility. The tradeoff in testing by developers is that the time required for testing is taken from the development of new functionalities. Talby et al. discovered that the traditional view of applying an independent tester is not sufficient in an agile development project. Testers working in isolation from developers resulted in an almost worthless approach, and management saw that developers could effectively test the software themselves. Testers working in close interaction with developers were seen as a more suitable approach for agile projects. However, Talby et al. did not provide any concrete results of this approach.

Sumrell [30] struggled with quality assurance activities when trying to move from the waterfall model to agile development. After discussions with agile consultants, Sumrell identified that unit testing done by developers is crucial for their project and that developers have to take responsibility for the quality of the product, as well. In that way, testers could focus on acceptance and functional testing. In order to successfully accomplish testing activities during the iterations, testing should start from the very beginning of the project [30]. In addition, the use of lightweight test automation tools is emphasized.

Test automation was seen as a key factor in agile testing in Puleio's [22] study. With test automation, it was possible to keep testing and development in synchronization. In Puleio's project, they applied Scrum and XP together to experiment with agile techniques. Their results state that one of the main challenges they faced was software testing; insufficient understanding of software testing raised questions inside the development team, while communication between stakeholders was the key to solving this matter. Finding a common language helped the team to communicate and understand the effort testing requires. Another issue was the estimation of the testing activities. Testers could not divide the testing tasks into appropriate pieces of work that would be possible to complete during the iterations. However, estimations improved during the development process, even though one of the testers did not approve breaking down the testing into smaller tasks.

In Shaye's [26] experience, test automation plays the greatest role in agile testing. Shaye describes an excessive test automation environment and strategy for agile development in her paper. Continuous testing would have been impossible to attain without test automation. With the right priorities, testing was targeted to high risk areas of the product. Testing conducted by developers was not emphasized, as in the studies of Talby and Sumrell. Testers were required to develop test automation and to implement more complex testing, e.g. exploratory testing. Testers and developers were paired to work together to provide quick feedback on the developed functions.

A recent study by Stolberg [28] supports the use of test automation to make testing faster in an agile environment. However, Stolberg also had problems with testing when applying agile methods. According to him, it was difficult to test in parallel with development. Again, with the help of test automation, continuous testing was made possible. The building of test automation required a large amount of consideration; however, it was worth the effort. In this case, acceptance and unit tests were automated. Unit tests were written by developers and acceptance tests by testers.

Itkonen et al. [12] studied the topic by comparing the practices in plan-driven methods and in agile methods. As seen from the presented studies, an iterative and fast development environment brings challenges to software testing. Itkonen et al. lists the challenges brought by agile principles and also the challenges that plan-driven testing principles introduce (e.g. independence of testing). They evaluated the agile quality assurance activities with the Cycles of Control (CoC) framework. This CoC framework illustrated in which time segment a quality assurance activity is positioned. Itkonen et al. suggested that introducing an independent tester could bring improvements to agile testing. However, they emphasized the need for more research on the topic in order to gain knowledge of whether additional testing practices are even needed in agile methods.

Based on these studies, it seems that test automation is a crucial part of testing in an agile environment. Another aspect is that testing practices used in plan-driven methods may not be compatible with agile processes. In addition, it seems that the role of testers is not as clearly defined in agile development as it is in plan-driven development. As for our study, these findings were used as a basis for qualitative research. Our objective was to observe how applying agile development, or agile practices, to the software process affects the testing process, and which the most prominent factors are that cause it.

3. RESEARCH METHOD

Software testing at the organizational level has several aspects, which can be considered to affect the effectiveness of the test process, and ultimately, the quality of the end-product. These seem to vary between different types of organizations and software projects, as even in the seemingly similar organizations the approach to the software and test process may differ significantly [16]. Some of this can be attributed to the human factors in the process. The testing work involves many actors, such as developers or testers, and communication between these actors, meaning that these factors should be addressed in observing the organizations [23].

3.1 Grounded theory

Acknowledging these conditions, we selected the grounded theory approach [10 and 29] to conduct an empirical analysis of the software testing practices. The grounded theory approach was considered suitable, as observing and describing organizations and phenomena are its strong areas. For example, Seaman [25] discusses grounded theory research as a way to identify new theories and concepts, making it a valid choice for research in software process and testing practices, and therefore suitable for our needs.

Grounded theory was first introduced by Glaser and Strauss [10] as a method for collecting data and creating theories in social sciences. The modern methodology has two main approaches: Glaser, as introduced in [9] and Strauss and Corbin, as introduced in [29]. In this study, we applied the Strauss and Corbin approach, which places a greater emphasis on the systematic categorization of observations and abductive reasoning [32]. In theory building, we followed guidelines by Eisenhardt [5], with additional input and principles for interpreting field study results derived from [18], [20] and [21].

3.2 Data collection

In our study, the focus was set on the level of organizational units (OUs), as described in the standard ISO/IEC 15504-1 [13]. The organizational unit is a part of an organization, deploying one process or more within a coherent process context, operating within set policies and objectives. Typically, an OU is one part of a larger organization, but especially in micro- and small-sized businesses, as defined in European Union SME definition [6], an OU consist of the entire corporation. In other definitions, such as TMMi [33], the definition of an OU is further elaborated. The daily activities and management of the processes in OUs are internal, but the activities are steered from upper level management with motivators such as corporate policies or operational strategies. However, the OUs usually have some ability, albeit a limited one, to affect these steering motivators, with activities such as enhancement proposals or feedback [15]. Overall, the OU was selected as an assessment unit because the use of an OU normalizes the company size and makes direct comparisons between different types of companies possible.

In our study, we interviewed 12 OUs, which represented different operating domains and polar types [5] of software industry. Our selection varied from small, national companies to large international software producers; from electronics producers to software houses. The amount of agile practices in the participating organizations varied greatly. One organization applied a complete SCRUM-based approach in software projects, while another had a completely design-based approach in software development. However, in almost all organizations some agile practices were applied; for example, explorative testing, iterative development cycles and feature set-based approach were common. The participants can be roughly divided to three groups based on their agility: low, medium, and high, based on the amount of applied practices. In the high level, an organization applies agile development method in all activities. In medium, the main development process may be design-based, but additional projects like feature development or updates, are done with agile approach. Also organizations that applied several agile practices were considered medium. Low is the level where only individual users or individual projects applied agile practices. From “low” organizations, three applied almost strictly design-based development processes. On the other hand, one organization was dedicated in introducing agile practices to their processes. A brief description of the OUs - and their main agile practices – participating in all of the data collection rounds, is available in Table 1.

The initial population was selected based on our prior research [15, 16 and 17] on with the combination of the polar type selection method, meaning that the organizations were selected to be as heterogeneous as possible and included as many different types of business activities in the software development business as possible. The participants were selected from our research partners, and supplemented with additional organizations to fulfill the polar type requirements. All of the selected organizations were professional software producers of a high technical level, producing software as their main activity.

Our objective was to gain insight and understanding of the software test process in selected organizations, and later analyze which concepts affected the test activities and how the test organizations worked. To approach this problem, we decided to interview several people from the organization during three interview rounds, in which we used semi-structured questions on

Table 1: Description of the Interviewed OUs

OU	Business	Company size[2] / Operation	Amount and types of agile practices in the organization
Case A	MES[1] producer and electronics manufacturer	Small / National	Low; explorative testing, focus on communication between stakeholders
Case B	ICT consultant	Small / National	Low; develops software as feature sets
Case C	Logistics software developer	Large / National	High; applies SCRUM [24]
Case D	Internet service developer and consultant	Small / National	Low; testing activities interweaved to development
Case E	Safety and logistics systems developer	Medium / National	Low to none; module reuse in non-critical aspects
Case F	Maritime software systems developer	Medium / International	Medium; applies iterative development cycles in most projects
Case G	Financial software developer	Large / National	Low to none; unit testing done by developers
Case H	ICT developer and consultant	Large / International	Low to none; unit testing done by developers
Case I	Financial software developer	Large / International	Low; piloting agile practices in development
Case J	SME[2] business and agriculture ICT service provider	Small / National	Medium; process able to adapt to the product, feature sets based on customer requests
Case K	MES[1] producer and logistics systems provider	Medium / International	Medium; daily builds, some SCRUM [24] activities, test automation
Case L	Modeling software developer	Large / International	Low; explorative testing

[1]Manufacturing Execution System [2]SME definition [6]

Table 2: Organization of interview rounds

Round type	Number of interviews	Interviewee role	Description
1) Semi-structured	12 OU interviews	Designer or Programmer	The interviewee was responsible for or had influence in the software design.
2) Structured and Semi-structured	31 OUs, including 12 OUs from 1st and 3rd round	Development or Testing Manager	The interviewee was responsible for a sofware project or a testing phase for a software product.
3) Semi-structured	12 OU interviews	Tester or Programmer	The interviewee was a dedicated tester or was responsible for testing the the software product.

themes such as the development method, testing tools, test automation, perceived quality aspects and such.

The data collection was completed in three rounds, in which we interviewed software designers, development or test managers and actual testers, one from each participating OU. Typically, the interviews lasted approximately one hour and were held face-to-face at a location selected by the interviewee, typically at their office. All of the interviews were tape-recorded for later analysis. If the organization did not have separate designers or testers, we interviewed the person, usually a programmer, whose responsibilities matched the desired role description. On two occasions, the OU selected two persons for an interview, as they considered that they did not have only one individual with sufficient knowledge on the interview topics. The interviewed persons were also allowed to see the questions beforehand and prepare for the interview if they deemed it necessary.

As a supplemental source of information, on the second round we also conducted a structured survey on the participating OUs. This was decided to gather also quantitative information on the test resources and current state of the overall test process. The quantitative survey was conducted in 31 OUs, all of the 12 interviewed organizations included. As for the grounded theory analysis, the answers to the qualitative questions in the second round from 19 additional OUs were discarded. The qualitative data was subsequently used as an additional source of information in the data analysis, in establishing relationships between different categories. A summary of the data collection rounds and participants is presented in Table 2. The interview questions, which included such topics as development method, test phases, testing tools, testing methods and quality concepts, can be found at http://www2.it.lut.fi/project/MASTO/.

We decided to start the interviews with designers in the first round to test our prior observations on the test process and to see if the results from our prior research project [15, 16 and 17] were applicable. In this way, we could also elaborate on the execution level details, further refining the questions for the latter interview rounds.

The managers were interviewed in the second round. During the interview we also collected survey data with additional structured questions. The decision to survey managers was made because they tend to have better understanding of the project- and organization-level aspects such as standards, testing policies and customer influence on the test process. As for the other estimates, the manager-level personnel were considered to be in a better position to assess such aspects as resource availability or personnel competence.

The third group that was interviewed was the testers. During this round, the focus was on testing activities and everyday testing. During this round, the same group of OUs was used as during the first round.

3.3 Data analysis

The grounded theory method contains three data analysis steps, which include open coding, axial coding and selective coding. During the first step, open coding, the observations from the interviews were codified to represent different phenomena and aspects that are related to the analyzed topic. The observations were connected to groups called categories, which lead to definitions of relationships and interconnection between categories.

The categories were developed from "seed categories" [5], which in our study were collected from our prior studies, literature review, and derived from the research question. For example, "Software development method", "Test resources", "Process problems" and "Role of the upper management" were a few of the applied seed categories. In practice, the open coding phase meant that the different observations were given the form of "category: phenomenon", such as "Test process: Applicability of agile methods" or "Test resources: Division of available resources", so that their relations and concepts connected to the categories could be further elaborated. During the coding, some of the prior categories merged, new categories were developed, and some categories altogether rejected by comparing different observations to find patterns, similarities or regularities with each other. After the open coding, our data set collected from 36 interviews resulted to 166 codes in 12 different categories.

In the axial coding, the focus was on the causal conditions and relations between the existing categories. In this phase, the categories themselves were examined more closely to establish any kinds of connections between them on the larger scale. At this stage, the observations and codifications were becoming rigid, allowing the focus to shift towards defining a connection between larger concepts such as categories.

The objective for selective coding is the definition of the core category [5, 29]. The core category may sometimes be one of the existing categories, but it can also be an amalgam of categories should none of the final categories be broad or influential enough. In this study, the core category was identified to be such an amalgam, consisting of the test strategy, the test organization, and the customer. Also the existing, known problems of the test process were considered important part of the study, as they pointed out possible difficulties in the process of applying agile methods and software testing. Basically, these observations meant that instead of one influential aspect, the core category in this study was identified to consist of several test process and software development issues, listed in Figure 1.

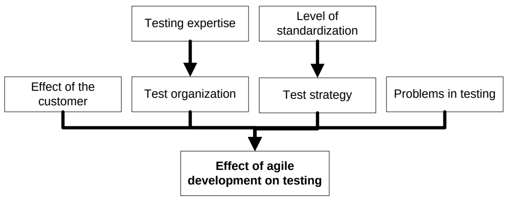


Figure 1: Central aspects of agile development that affect the testing work

Overall, the core category is the "Effect of agile development on testing", with categories such as the effect of the customer, test organization, test strategy and problems of the existing testing process being the major components, which define the connection between agile practices and the test process, and are explained with a hypothesis derived from the observations.

4. RESULTS AND OBSERVATIONS

The hypotheses of the study were derived based on guidelines of constant comparison, discussed in [20]. The observations were compared to each other and joined, delimited and written up as prominent regularities or patterns and used to establish main categories for the study. Based these categories, and group of observations, the hypotheses were derived.

To limit the number of hypotheses, and to find the most prominent hypotheses from the data, we applied a simple development and elimination system: first, we derived a total of ten propositions as candidates from the observations. We then discarded or combined these propositions together based on their observed relationships. In the second round, we derived four new propositions, and combined or used them to reject existing candidates. This way, we were able to discuss or explain the observed patterns and regularities from the data. Finally, we encapsulated propositions as five hypotheses.

4.1 Categories

In this study, our focus was on the practical impact on testing activities. The purpose of the categories is either to explain the phenomena observed, or to differentiate the case organizations based on testing or software process activities. Usually the observation was made from the interview data, but for some concepts, such as the customer role or the effect of agility on the software process, the relationship was established in related research and taken into our study to confirm the results. From the data, we observed six major categories which could be used to differentiate the organizations based on their agile practices and test processes. A summary, along with a short description of the developed categories, is also available in Table 3. Overall, the following categories were created:

The category *test organization* explains how organization units are structuring their testing and who is responsible of testing activities. In this study, testing was organized diversely in different companies. Even though we studied the organizations at the level of organization units, differences between large and small companies were visible. In small companies, the responsibility for testing belonged to developers, and in few cases, testing was partly assigned to customers. In medium-sized companies, testing activities were clearly assigned. However, from three medium-sized companies only one had dedicated testers. In large companies, testing was mainly structured in a way that testing had its own unit or team; only one organization unit belonging to a large company did not have dedicated testers.

The category *testing expertise* describes which kind of knowledge is important for a tester to possess in the organization unit. Generally, domain knowledge was the most emphasized area of expertise for testers to have in the studied OUs. In a few cases, interviewees thought that testing without any knowledge of the application area would have been very difficult, and therefore, they did not use inexperienced testers from the domain in question or outsourced testing. Technical knowledge was seen important, as well, because testers should have a basic understanding of techniques to successfully execute testing. In an organization unit where Scrum was applied, technical knowledge was considered to be more important than domain knowledge. A tester working in a Scrum team considered that testing in an agile project is more difficult than in a project which follows the iterative waterfall model.

The category *the level of standardization* describes the types of processes and practices followed in the OU. For OUs studied in this research, it was typical that OUs followed practices they had discovered within the organization. Official standards and capability-maturity models, such as the ISO 9000 series and CMMI, were rather guidelines or a basis for the OU's own rules and practices. Only one OU followed the ISO 9001 standard strictly in their development process. Again, differences between OUs belonging to small organizations and OUs belonging to large

Table 3: Categories defining testing and agile development

Category	Definition
Test organization	The personnel roles and structure of the testing organization.
Testing expertise	The primary knowledge which is required for a tester to have.
Level of standardization	The types of formal processes and practices followed.
Effect of the customer	The role of a customer and how a customer is involved.
Test strategy	An approach and the primary methods of software testing.
Problems in testing	Issues and hindrances that the testing organization is facing.

organizations were visible. In small companies, guidelines and practices were observed if the practices were in-house standards. Generally, the in-house practices for development and testing in medium-sized and large companies were based on official standards and they were also followed. The OU that used Scrum was an exception in that it could not use standardized processes because they were not compatible with the fast development cycle. The project manager of the OU said that they had to design their own processes especially for testing in an agile environment.

The *effect of the customer* category describes the effect the customer has on the OUs. In two OUs, the customer did not participate in the development or testing of the product. However, both of the OUs documented the feedback they received from customers in order to improve their product. In other cases, customers were engaged in the development or testing at least at some level. Common approaches to customer participation were that in testing customers performed acceptance testing and in development they attended to requirements gathering and definition. In one of the OUs, customer relationships were considered to be very important since the OU turned the responsibility for testing over to customers. In the studied OUs, it seemed that internal customers were involved in the development process more closely than external customers. Overall, close customer participation was desired.

The category *test strategy* explains the approaches the OUs took to software testing. A common approach in the studied OUs was that testing focused on the main functionalities or on assuring that the system operates as it should. Consequently, testing resources were concentrated to test the most critical parts of the software. Most of the OUs belonging to large organizations based their testing on test plans and the test policy. An approach the OU using Scrum took to testing was to test everything and with speed; however, if optimizing was needed, risk based testing was utilized. In addition, instead of performing exploratory testing, which in some literature is considered to be an important part of agile testing, they made test plans which they followed. OUs belonging to small organizations did not have a plan for testing, and the observation of a test strategy was up to the developers or a project manager. Medium-sized companies had test plans and a common test strategy described above. However, testing activities were poorly monitored.

The last category includes the *problems in testing* in the OUs. The OUs following the traditional plan-driven processes had rather traditional issues in testing, as well. A common issue was the absence of test resources, such as time and personnel. A problem that occurred in the OUs that did not have dedicated testers was that when testing was the additional task of a person, testing tasks were bypassed if there was not enough time to successfully perform both testing and the primary tasks. The OUs belonging to small organizations had problems with the test strategy and test plan. A common cause for these problems was that development tasks were of higher priority than testing tasks. Hence, the planning and execution of test activities were omitted. Issues that the OUs belonging to larger organizations faced were diversified. An agile OU had an issue with the absence of documentation, and another OU using the plan-driven method had difficulties in sharing knowledge by direct conversation. None of the representatives of the OUs believed that they had 100 percent of the needed test resources.

4.2 Hypotheses

We created the hypothesis based on the categories described earlier and observations from the organization units. The hypotheses were shaped to generalize or explain the observed similarities or differences, following the chain of evidence from study concepts to case observations and interviewee opinions. For example, the hypotheses 1 and 2 were derived from observations when comparing the different organizations, and the interviewee opinions on how the agile practices affect – or would affect – their testing work. Similarly, hypotheses 3 and 4 were summarized from two larger categories of observations; problems of test processes and customer participation. Fifth hypothesis was conceived from interviewee comments regarding benefits from applying agile practices. The observations, which were used as a basis for these hypotheses, are also available in summarized form in Table 4.

4.2.1 Hypothesis 1: Agile practices tend to allow more time for testing activities, while the total time for the project remains the same.

Agile practices reserve more time for testing and speed up the project by dividing and delivering the developed product in pieces. This approach enables the implementation of the most critical parts of the software first, which allows testing to start early and focus on the critical parts first. Testing can be executed piece by piece during the development; hence the completion of the entire software is not required in order to start testing.

”I believe we could then deliver the software in smaller shipments and could get the most critical parts done much faster.” –Tester, Case G

In plan-driven methods, testing tasks may be discarded or downscaled if the implementation and specification phases require extra time. Especially if there are no separate testers, the developers are obviously needed primarily for building the required features. In most cases, testing cannot extend the deadline, and if testing is the last phase, there might not be enough time to correct all defects found in the end of the project, as extending deadlines may be a costly or entirely unfeasible option.

”Being involved already in the project planning stages is really good. I have much deeper knowledge of what I’m testing when I get to testing it.” –Tester, Case C

”How an agile method affects testing is that it directs the testing to be performed during the project, that testing isn’t just the last matter of the project” –Designer, Case B

When comparing agile methods to plan-driven methods, the extra time for testing is explained by the early involvement of testing. Testers have a better insight into the product, since they participate in the project earlier. Early testing enables quick feedback on the product to the developers. Furthermore, there may be more time to correct certain types of defects, as they may be found early in the development, not only at the end of the project.

”It affects so that testing is performed for smaller pieces and testing is done earlier. In our earlier process, testing started with great speed after development work was finished and then the defects were found too late. I think it is a good thing, since it should make bugs visible earlier.” –Tester, Case F

4.2.2 Hypothesis 2: Applying agile practices smoothens the load of test resources, but does not reduce the total amount required during the entire project.

Delivering the product piece by piece enables the testing to start early. This approach may even be a more effective solution for executing tests, since the targets of the testing process are focused on certain parts of the software. These parts are tested throughout the project, and used later on as a basis for additional tests. Testing is also done continuously during the project, which eases the required effort and need for test resources when compared to plan-driven methods. The need for test resources is not reduced because test resources are needed during the course of the project.

"It changes it so that resources are needed continuously, not just once or twice a year." –Tester, Case F

"If the end product is in pieces, it in some sense reduces the amount of work, since there is always a certain restricted part to be tested. And on the other hand, testing a part at a time, possible defects are better detected." –Tester, Case B

"It sort of levels the testing, because now testing takes place in chunks. When the product is ready, suddenly there are two or three weeks of time to test it and then testing is done 24/7. So, it

Table 4: Overview of the observations from Case OUs

Case OU	Test organization	Testing expertise	Level of standardization	Effect of the customer	Test strategy	Problems in testing
A	Developers	Experienced developers	No standardized processes	Partly involved in development, does acceptance testing	New functions and changes, exploratory testing	No strategy or plans, resources (time, personnel)
B	Project manager or customer, customer service	Person specific, domain knowledge	Official, ISO 9001, not strictly followed	Participate in testing, relationship is important, close collaboration	New and core functionalities, explorative, resources are focused	Plan, strategy, resources (time, personnel)
C	Dedicated testers working in a Scrum team	Senior-level testers or test consultants, technical knowledge over domain knowledge	Company-specific practices that are followed variedly	Internal customer, close collaboration in development	Test everything or risk-based, rely on test automation, resources are focused, follows test plan	Documentation, senior-level testers required, resources (time, personnel)
D	Field testing	Users' know-how	Company-specific processes, no process for testing	Customer is not involved in development or testing	Core functionalities and critical targets, separate test projects	Plan, process, costs, resources
E	Developers and designers	Designers (testers) technical and domain knowledge	Company-specific processes, based on ISO 9001	Involved in testing and specification of requirements	Core functionalities, operation, coverage, plans and surveillance, test automation	Plan, surveillance is weak, resources (time, personnel)
F	Internal customer	Domain knowledge	Company-specific processes	Internal customer, arranges testing, close collaboration	New and core functionalities, explorative, resources are focused, test automation	Test automation, do not hire dedicated testers, resources (time, personnel)
G	Separate unit for testing	Testers' domain and technical knowledge and experience	Company-specific processes and policy, based on ISO 9000 –series, strictly followed	Internal customer, involved in development, surveys testing	New and core functionalities, operation, test policy, plan and surveillance, test automation	Maintenance of test automation, communication, inconsistent division of resources
H	Separate workgroup for testing	Domain knowledge	Company-specific processes, based on CMMI	Does acceptance testing, possibility to define tests	Functioning, resources are focused, test automation	Test coverage, plan, surveillance, resources (time, personnel)
I	Customer service, project manager	Testers domain knowledge, experience of project manager	Own internal practices and guidelines are followed, no process for testing	Feedback is important, no other involvement in development work	New functionalities, operation, coverage, project manager directs the testing, explorative.	Strategy, development of test automation, resources (time, personnel)
J	Separate unit for testing	Testers' domain and technical knowledge, and testers should know organizational practices	Official, ISO 9001, strictly followed	Accepts specifications, participates in acceptance testing, surveys the progress of testing	New functionalities, operation, coverage, follows standard, plan, surveillance	Strategy when testing new, resources (time, personnel)
K	Separate unit for testing	Domain knowledge	Company-specific processes, based on CMMI, are strictly followed	External customer accepts test plan, internal customer surveys development and testing	Operation, test policy, plan, some explorative.	Resources (not enough test environments), testing changes
L	Testing director	Testing directors' experience, domain knowledge of staff	CMMI, company-specific process for testing is obeyed	Internal customer, close collaboration, is involved in specification, surveys testing, does acceptance testing	Operation, coverage, stress, plan of the testing director, surveillance, test automation	Plan, strategy, no professional tester, need for tacit knowledge, documentation practices not followed

could level the load of test resources." –Tester, Case G

However, if the need for test resources is examined on a daily basis, the need may vary.

"On the agile side, the need for test resources may vary in the short term so that resources are needed in bursts. It may be for example that some day several testers are needed, but on the next day testers might not be needed at all." –Tester, Case C

Testing can be more efficient if testers can focus on one project at a time. In case C, the OU achieved efficient testing by allowing the testers to work only on one project at a time. A tester from the organization unit C thought that they are more efficient because they do not have to change between different projects.

"What we have noticed is that, before, when we were using traditional methods, a tester might have been spread across two or three different projects, and it was quite difficult because you spent so much time changing what you were working on that you were a lot less efficient. Now testers are usually dedicated just to one project." –Tester, Case C

4.2.3 Hypothesis 3: In order for the development and subsequently testing to be effective, the stakeholders have to understand and conform to the practices in agile methods.

In order to use agile methods, the vendor organization is required to increase the amount of collaboration and communication with the customer. Every stakeholder should understand the practices applied in agile methods. An agile development process requires a different approach to development from the customer point of view, as well. Requirements and specifications are not strictly defined at the beginning of a project; they are allowed to change and develop gradually during the development process. Customer participation is important throughout the project, since requirements are clarified during the course of development. In testing, the close collaboration shows in quick feedback and maybe in more effective acceptance testing, since testing can be performed gradually, as well. Another aspect is that old testing methods may turn out to be insufficient when agile practices are observed. This may lead to the OU being forced to think of the testing practices and activities that should be utilized in agile process.

"Maybe not a process, but we have certain practices of our own especially for testing in agile methods, since the traditional testing methods do not really fit the agile process." –Project manager, Case C

"We have customer projects which have strict contracts and instructions, and because of these strict contracts we are forced to conduct the implementation so that the customer defines the specifications. If it doesn't go well, lawyers will deal with it. So, in this kind of situation we can't use agile methods." –Designer, Case F

"We don't really use agile methods, which is because of our customer interface. It is difficult to find customers who want to be involved in an iterative development model. Most of the customers want a package solution with a package price." –Designer, Case H

"I believe it would require a new style of thinking from us and from the customer so that we could understand it correctly." – Tester, Case L

With the increased collaboration and communication, the feedback and opinion of the customer is received quickly. Acceptance tests can be more efficient because when delivered in pieces, testing a part of the software is faster than testing the whole software at once. This kind of approach makes it possible for the customer to test pieces of the product throughout the project and give more precise feedback about each piece. The participation of a customer also helps the developers to develop a better vision of what the customer truly expects and wants from the product being developed.

"In my opinion it works better then. Especially when making new software, the customer gets the software piece by piece and if the customer is not happy with some part of it, he can say so immediately." –Tester, Case B

To make the development process effective, conformance to the agile practices is essential.

"The first feature the customer wants should be implemented first, but in our case it was done last because it was the most difficult feature. My belief is that the most difficult features should be done during the first sprints." –Tester, Case K

"Let's say that in the inception phase the Scrum process usually starts very sluggishly and the first sprints are the most difficult ones, but it gets easier in later sprints." –Tester, Case C

4.2.4 Hypothesis 4: Internal customer supports the deployment of agile methods.

In the principles of agile methods, the significance of customer collaboration is emphasized. In the studied OUs, the collaboration was considerably closer if the OU had an internal customer. For an internal customer, it is easier to participate in the development work continuously. It may be challenging to tie an external customer to the project, since agile methods require close collaboration throughout the project. In addition, they require reliance between the external customer and vendor in order to use agile methods without fear of serious disagreements.

"We have a lot of internal customers. And most of them participate more than you would find in most other organizations, actually." –Tester, Case C

"Then there are certain customers, with which we have attained a good relationship, and we can use agile methods with them." – Designer, Case B

The internal customer and vendor should have a good relationship and mutual reliance. This enables the development work to be more flexible. A good relationship increases the overall understanding of a customer about the developed product and about the possible issues during the development process. Communication with an internal customer should be more convenient, since the people know each other already.

"Our operation is quite consistent, so a positive matter is that usually the development team and representatives of the customer know each other already. They don't need to find a common language, since it is already established." –Designer, Case L

4.2.5 Hypothesis 5: Applying agile methods allows faster reaction times for change and feature management in testing.

In case I, the developer wanted to make changes to the product at short intervals, but still maintain the product in a deliverable state. In agile methods, the length of the development iteration can be set to fit the organization and the change. As testing is part of the iterations, and not just a phase at the latter stages of the project, the application of agile practices allows testing to be more flexible in case changes are needed.

"Some of the developers try to make smaller changes that can be finished in certain period of time, so the product could stay longer in one piece. Especially when the situation of a project is stable, we like to keep the product deliverable at all times." –Designer, Case I

Organization A applied practices that are typical for agile methods in the projects where they updated the existing product, although they formally followed the plan-driven approach in major development projects. The OU collected customer requests and analyzed them for implementation features.

Overall, it seems that agile methods do offer some advantages in terms of correctness and appropriateness, since the progress can be evaluated and the aim of the development corrected at certain intervals. Design faults or origins for persisting errors can be tracked and reacted on, translating into easier testability of the software.

"And when we have gone on to develop totally new things, it hasn't been easy. We would have needed a different development model or something." –Tester, Case J

"When you start to develop some kind of software with agile methods and you slip away from the right track, it is faster to find and deal with the problem you are facing." –Designer, Case K

4.3 Supplemental observations

Generally, in the studied OUs, it seemed that testing tasks were seen as less important than development tasks, and that several text-book hindrances for the test process existed. In two of the OUs, dedicated testers were not hired because management considered that testing could be performed sufficiently by developers during the development phases. Furthermore, in some cases the organization had dedicated testers, but reserved too little time to perform all of the necessary testing activities during a project. However, Case C was an exception because the organization developed its testing methods instead of downsizing the test set if shortage of time caused problems. Testers constantly looked for quicker ways to execute test sets rather than leave cases out of the test plan.

5. DISCUSSION

The results of this study consist of five hypotheses derived from the material of the study. Our objective was to explore how agile methods affect the execution of software testing in practice.

We observed that agile methods have a tendency to require extra time for software testing in a software project. This can be achieved by delivering, and developing, the software in smaller iterations, giving more time to execute testing tasks within each of the iterations and by allowing an earlier start to the testing work. As software testing can be started early and is completed concurrently with the development work, it can focus on performing the tests for the software piece by piece, as well, and elaborate on the already existing parts of the final product instead of being simply the last project phase before delivery.

The study by Itkonen et al. [12] supports the observation made of the early involvement of testing in agile methods. In their paper, Itkonen et al. used the CoC framework to describe how testing activities are located in an agile process, and established that even though there is only a limited amount of literature or studies in relation of agile methods and testing, agility can emphasize certain quality-building practices. However, the early involvement of testing brings forth new issues, as Puleio [22] and Stolberg [28] have noticed. Even though testing can start early and be executed concurrently with development, there are difficulties in how testing should be deployed in the iterations.

A second important observation was that agile methods have a tendency to make demands for test resources more predictable. While still retaining the same needs for test resources, this can be considered advantageous especially in organizations where test resources are limited and competitively distributed between projects. In agile methods, testing becomes part of the project in the early stages, so testers are also a part of the development team for most of the process. As testing is carried out throughout the project, this causes a continuous need for test resources. Continuous and early participation in testing allows extra time for the execution of tests, and sets a clear focus for the test effort to concentrate on a certain subset of the software. This can be used to prevent cases, where testing tasks become a burden for the development team in the latter parts of the software project.

However, this has aroused some discussion over the role of testers in agile processes. For example, Talby et al. [31] question the need for testers if testing is carried out as a natural part, or an extension, of developers' work. However, the result of this study implies that dedicated testers are needed in agile projects, as there still exist test phases where separate testers are beneficial. In addition, these results are in line with another study by Shaye [26], in which testers were paired with developers. This enhanced the output by generating quick feedback for the developers.

Other results of the study were closely associated with customers and collaboration between the customer and vendor organization. Agile methods rely on the presence of the customer and collaboration with the customer. The role of the customer is more collaborative than in plan-driven methods. Hence the agile process requires the customer to adapt to closer collaboration. It has been established that increased communication between the customer and vendor organization should make it easier for both parties to understand the development project [4 and 24]. Through close collaboration, customers can increase their understanding of software development, while the presence of the customer may be helpful for developers as well. In this case the customer can specify unclear requirements or functions directly to the developers, who then can design requested changes into latter development iterations.

In this type of research project, there also exist several threats to study validity [34]. For example, in codification of observations the researcher bias can be troublesome, skewing results on data analysis. Similarly, design issues on questionnaire could have steered the collected data towards certain viewpoints. In our study, the threats to validity were addressed by taking certain measurements to ensure neutrality. For example, the questionnaire was designed by group of four researchers, with feedback and adjustment ideas collected from other software engineering researchers. The interviews were conducted by the questionnaire designers, to ensure that the interviewees understood the questions correctly. Finally, the codification process was conducted by four researchers, two of which did not participate on the design or interview process, to ensure minimal interference of personal opinions or individual preferences.

Another concern was the number of agile organizations. This research was limited to studying a selected sample of 12 organization units. One limitation of the results was that in the studied OUs, only one organization applied a fully agile development process in its main development projects. However, most of the other organizations applied some agile practices, or used agile development as a secondary development method for updates or feature development projects. One organization was in the process of introducing agile development into their projects, while two other organizations applied several agile practices. Even though there were some cases where agile methods were not used for varying reasons, such as end-product criticality, customer requirements or business domain requirements, all organizations had some experiences in agile practices. This can be seen as an indication that purely agile methods are not applied as widely as literature would suggest, but that many organization have adjusted their processes to accept “best practices” from agile development concepts.

6. CONCLUSIONS

In this paper, we presented our research on test processes in software organizations. Our study observed software development processes which apply agile methods or agile practices and examined how these activities affect the test process when compared to test processes based on the traditional plan-driven approach.

Based on Sumrell’s [30] study, agile methods are usually applied in the software process to find out what the customer truly expects from the product, to speed up the entire development project and to include testing activities into the project from the very beginning to help predict the required testing time. Based on the observations made in our study, it seems that the software organizations which apply agile methods are in a position to achieve the goals Sumrell has suggested. It seems that organizations which apply agile methods are generally more flexible in terms of changes and testing in the software process. However, testing in parallel with development work is difficult to execute and requires a reasonable amount of consideration from the organization. It could be stated that to successfully incorporate testing into agile methods, the development organization is forced to think and revise their testing processes in detail.

Our objective was to define how agile methods affect the test process of software projects. Based on our observations, it seems that there are some benefits, such as the early involvement of testing, which may arrange more time for the execution of testing, and simultaneously makes the need for test resources more predictable. On the other hand, agile methods expose the software process to such hindrances as the lack of test documentation, which can lead to problems when transferring testing tasks to an outside organization or using testers who have limited experience in the product environment.

The definition of how agile methods affect the test process is valuable in developing the software organization, as it enables the test process to address issues in advance. This information may become crucial when defining a new test strategy or steering the organizational software process towards agile development methods. In our future work, our objective is to develop these observations and offer guidelines on how agile development methods should be addressed in a test organization and within its organizational test strategy.

7. ACKNOWLEDGMENTS

This study was supported by the ESPA-project (http://www.soberit.hut.fi/espa), funded by the Finnish Funding Agency for Technology and Innovation, and by the companies mentioned on the project web site (http://www2.it.lut.fi/project/MASTO/).

8. REFERENCES

[1] Ambler, S. 2008. Agile Software Development at Scale. *Balancing Agility and Formalism in Software Engineering,* 5082, 1-12, Springer, Berlin. http://dx.doi.org/10.1007/978-3-540-85279-7_1

[2] Beck, K. 2000. *Extreme Programming Explained: Embrace Change,* Addison-Wesley.

[3] Bertolino, A. 2007. Software Testing Research: Achievements, Challenges, Dreams. *Future of Software Engineering, 2007 FOSE '07*, 85-103.

[4] Boehm, B. and Turner, R. 2004. *Balancing Agility and Discipline: A Guide for the Perplexed,* Addison Wesley, Boston.

[5] Eisenhardt, K.M., "Building theories from case study research”, Academy of Management Review, vol. 14, pp. 532-550, 1989.

[6] EU, "SME Definition," European Commission, 2003.

[7] Fitzgerald, B., Russo, N. and Stolterman, E. 2002. *Information Systems Development – Methods in Action,* McGraw-Hill, London.

[8] Glas, M. and Ziemer, S., “Challenges for agile development of large systems in the aviation industry”, Proc. 24th ACM SIGPLAN conference companion on Object-oriented programming systems, languages and applications, Orlando, Florida, USA, pages 901-908, 2009, http://doi.acm.org /10.1145/1639950.1640054

[9] Glaser, B.G. “Constuctivist Grounded Theory?”, Forum: Qualitative Social Research (FQS), Vol 3(3), 2002.

[10] Glaser, B. and Strauss, A.L. , The Discovery of Grounded Theory: Strategies for Qualitative Research. Chicago: Aldine, 1967.

[11] Highsmith, J. and Cockburn, A. 2001. Agile Software Development: the Business of Innovation. *Computer,* 34(9), 120-127. http://doi.ieeecomputersociety.org /10.1109/2.947100

[12] Itkonen, J., Rautiainen, K. and Lassenius, C. 2005. Towards Understanding Quality Assurance in Agile Software Development. *Proceedings of the International Conference on Agility (ICAM 2005),* 201-207.

[13] ISO/IEC, ISO/IEC 15504-1, Information Technology - Process Assessment - Part 1: Concepts and Vocabulary, 2002.

[14] ISO/IEC, ISO/IEC 29119-2, Software Testing Standard – Activity Descriptions for Test Process Diagram, 2008.

[15] Karhu, K., Repo, T., Taipale, O. and Smolander, K., “Empirical observations on software testing automation”, Proc. 2nd IEEE Int. Conf. on Software Testing Verification and Validation, Denver, USA, 2009.

[16] Kasurinen, J., Taipale, O. and Smolander, K., “Analysis of Problems in Testing Practices”, Proc. 16th Asia-Pacific Conference on Software Engineering, 1.-3.12., Penang, Malaysia, 2009.

[17] Kasurinen, J., Taipale, O. and Smolander, K., “Software Test Automation in Practice: Empirical Observations”, accepted to Advances in Software Engineering, Special Issue on Software Test Automation, in press, 2010.

[18] Klein, H.K. and Myers, M.D., "A set of principles for conducting and evaluating interpretive field studies in information systems”, MIS Quarterly, vol. 23, pp. 67-94, 1999.

[19] Krasteva, I. and Ilieva, S., “Adopting an agile methodology: why it did not work”, Proc. 2008 International Workshop on Scrutinizing agile practices or shoot-out at the agile corral, Leipzig, Germany, pages 33-36, 2008. http://doi.acm.org /10.1145/1370143.1370150

[20] Locke, K., 2001. *Grounded Theory in Management Research*, SAGE Publications Ltd.

[21] Pare´, G. and Elam, J.J., “Using case study research to build theories of IT Implementation”, IFIP TC8 WG International Conference on Information Systems and Qualitative Research, Philadelphia, USA, 1997.

[22] Puleio, M. 2006. How Not to Do Agile Testing. *Proceedings of AGILE 2006 Conference (AGILE’06).*

[23] Robson, C., Real World Research, Second Edition. Blackwell Publishing, 2002.

[24] Schwaber, K. and Beedle, M. 2002. *Agile Software Development with Scrum*, Prentice-Hall, Upper Saddle River, NJ.

[25] Seaman, C.B. , "Qualitative methods in empirical studies of software engineering", IEEE Transactions on Software Engineering, vol. 25, pp. 557-572, 1999.

[26] Shaye, S. 2008. Transitioning a Team to Agile Test Methods. *Agile 2008 Conference (AGILE ’08),* 470-477.

[27] Sommerville, I. 1995. *Software Engineering*, 5th edition, Addison Wesley.

[28] Stolberg, S. 2009. Enabling Agile Testing Ghrough Continuous Integration. *Agile Conference 2009 (AGILE ’09),* 369-374.

[29] Strauss, A. and Corbin, J. 1990. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques,* SAGE Publications, Newbury Park, CA.

[30] Sumrell, M. 2007. From Waterfall to Agile - How does a QA Team Transition? in *AGILE 2007*, 291-295.

[31] Talby, D., Keren, A., Hazzan, O., and Dubinsky, Y. 2006. Agile Software Testing in a Large-Scale Project. *Software, IEEE* 23, no. 4, 30-37.

[32] Thagard, P. and Shelley, C., 1997. “Abductive reasoning: Logic, visual thinking and coherence”, *Logic and Scientific methods*, pages 413-427, Dordrecht: Kluwer.

[33] TMMi Foundation, Test Maturity Model integration (TMMi) reference model, Version 2.0, 2009.

[34] Onwuegbuzie A. and Leech N.L., “Validity and Qualitative Research: An Oxymoron?”, Quality and Quantity, Vol 41(2), pages 233-249, 2007. DOI: 10.1007/s11135-006-9000-3

Publication IV

Test Case Selection and Prioritization: Risk-based or Design-based?

Kasurinen, J., Taipale, O. and Smolander, K. (2010), Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 16.-17.9.2010, Bolzano-Bozen, Italy. doi: 10.1145/1852786.1852800.

Test Case Selection and Prioritization: Risk-Based or Design-Based?

Jussi Kasurinen, Ossi Taipale and Kari Smolander
Lappeenranta University of Technology
FI-53851 Lappeenranta, Finland
+358 400 213 864
jussi.kasurinen | ossi.taipale | kari.smolander @lut.fi

ABSTRACT

The objective of this qualitative study was to observe and empirically study how software organizations decide on which test cases to select for their software projects. As the software test processes are limited in resources such as time or money, a selection process usually exists for tested features. In this study we conducted a survey on 31 software-producing organizations, and interviewed 36 software professionals from 12 focus organizations to gain a better insight into testing practices. Our findings indicated that the basic approaches to test case selection are usually oriented towards two possible objectives. One is the risk-based selection, where the aim is to focus testing on those parts that are too expensive to fix after launch. The other is design-based selection, where the focus is on ensuring that the software is capable of completing the core operations it was designed to do. These results can then be used to develop testing organizations and to identify better practices for test case selection.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management - *Software quality assurance (SQA)* D.2.8 [**Software Engineering**]: Metrics - *Process metrics*

General Terms

Management, Design, Human Factors.

Keywords

Software testing, Test case selection, Empirical study, Grounded theory.

1. INTRODUCTION

In the software industry, launching a new product in time makes a big difference in expected revenue [9]. However, before its launch, the software has to be tested, which itself is a costly process that can yield over half of total development costs [18]. In addition, regardless of the investment in testing, it cannot cover everything as the size and complexity for achieving full-coverage testing increase almost exponentially when the size of the tested software product increases [36]. Therefore, in most software projects, the matter of selecting which test cases should be included in the test plan exist [26]. In reality, the number of test cases which can be used in test process depends on testing resources like personnel and schedule [32], while trying to maximize the testing output to enhance product quality.

Testing practices seem to suffer from several hindrances, like using shortcuts, reducing test time, poor planning and poor testability [7]. The attitude towards testing, culminating in the “Let go – deliver now and correct later” mentality, causes additional expenses that could be avoided with some reasonable investments [33]. In literature, test case selection is considered an important aspect of the test process, actually being one of the central aspects in building and defining test strategy [13, 34]. As limited test resources are usual in practice [32], there has to be some method of deciding what test cases are executed.

In this paper, we studied different approaches on how real-world software producing organizations select their approach to test case selection. Our approach was to apply the grounded theory research method [8, 31], observe the practices of different polar types of companies, identify how companies select their test cases, and explain why they apply this type of approach.

This study continues our studies on software testing practice. Our prior studies have covered such topics as process problems and enhancement strategies [17], testing resources and test automation [15] and outsourcing [16].

The paper is organized as follows: In Section 2 we introduce related research concepts and in Section 3 the approach that was applied in this study. In Section 4 we present our results and their implications are discussed in Section 5. Finally, the paper is closed with conclusions in Section 6.

2. RELATED RESEARCH

The selection of test cases based on costs or related risk is not a novel concept. For example, Huang and Boehm [9] discuss cost evaluation methods for testing. By ranking the test cases based on their value, i.e. the amount of money lost if the test fails, a 20% investment in testing is sufficient to achieve 80% of the software value. Similar results of testing cost-effectiveness have also been reported by Yoo and Harman [38]. Petschenik [25] even argues that testing can be organized effectively even with as low as 15%

of the perceived resource needs, if the resources are focused on critical aspects.

Redmill [26] discusses this concept even further. As complete testing is not possible, it directs the testing process towards selective testing. As for the test case selection approach, there are different methods, which vary in applicability or results, but in general the testers seem to agree on applying risk-based selection [1, 26]. However, the criterion on which the selection is based on is usually incomplete or undefined. This often leads to a solution where risk analysis is based on individual experience and can be biased. For example, for developers the priorities for technical risks may be well-adjusted. However, risks associated to other stakeholders, concepts like legal costs and compensations, loss of reputation for the company or maintainability by third party associates, are probably beyond the scope of a single software developer [26]. A study by Do and Rothermel [4] suggests that ultimately the selection and testing cut-off point is a tradeoff between the costs of applying additional testing versus the costs of missing errors. Therefore it is plausible in real life to cut testing short to keep the deadline, as the loss caused by product delay supersedes the losses caused by releasing error-prone software [9]. Only in such extreme cases as an incomplete implementation of core features or crippling quality issues delaying the deadline can be considered a feasible option [27].

One proposal to help the test process is to provide better testability for product components [17, 20, 21, 35]. The rationale for this action would be that supporting testability would speed up the process of creating test plans and allow easier test case generation. By having clearer objectives for testing and an easier way to ensure test coverage, the effectiveness of testing work could be increased without severe expenses [38]. However, this approach is not as straightforward and easily implementable an improvement as it would seem. In these types of projects, the test strategy becomes crucial, as the enablers for testability have to be implemented into the source code simultaneously with the actual development process. In other words, the developers have to plan the development ahead to make sure that every needed case can be tested [21]. Within software projects, this would require rigid plan-driven development or continuous testability analysis for verification purposes, which would obviously generate other expenses [21, 35]. In contrast, in some cases like in software product line development, the testability requirements and possibility for conformance testing are emphasized [20].

Software development methods are geared towards producing quality in software products [7]. For example, international standards like ISO 25010 [12] define quality as an amalgam of eight attributes like reliability, operability or security. In addition to these definitions, real-life measurements like the mean time between failures [9] or number of errors found in testing versus errors found after release [25] may also be used as indicators for software development quality.

Organizational testing practices may also vary because of other aspects, such as the development method, resources and customer obligations [14, 17, 32]. Even if the purpose of testing is to verify functionality and to increase product quality [7], practical applications do vary, as different approaches to software development allow different types of tests in different phases. For example, developing software with agile development methods differs from the traditional plan-driven approach to the degree that they can be seen as exclusionary to each other [2]. On the other hand, several techniques like pair programming [10], code reviews [22], think-aloud testing [23, 30] or explorative testing [4] have been developed to enhance product quality and ultimately make the testing process easier. Even the task of generating test cases from which the selection is made varies; for example, black box testing and white box testing define two approaches to case generation based on knowledge regarding the structure of the object being tested [34]. However, these types of approaches focus on the generation process itself, not actually on defining how the test cases are selected, and in case of resource shortages, on the decision of which cases to include and exclude.

Overall, there seems to be an abundance of information and studies regarding test case selection in regression testing [e.g. 3, 4, 28], with several different models for cost/benefit-calculations and usability assessment methods. However, there seems to be lack of studies in software development, where regression and conformance testing models are not applicable.

3. RESEARCH METHODS

Software testing is a complex phenomenon, which has several related concepts and different approaches even with seemingly similar organizations [17]. Acknowledging this, we decided to pursue empirical qualitative analysis by applying the grounded theory method [8, 31]. Grounded theory was considered suitable for discovering the basis of testing activities, as it observes and describes real-life phenomena within their social and organizational context. According to Seaman [29], a grounded approach enables the identification of new theories and concepts, making it a valid choice for software engineering research, and consequently, appropriate to our research.

Our approach was in accordance with the grounded theory research method introduced by Glaser and Strauss [8] and later extended by Strauss and Corbin [31]. On the process of building a theory from case study research, we followed guidelines as described by Eisenhardt [5]. The interpretation of field study results was completed in accordance with principles derived from [19] and [24].

3.1 Defining measurements

The ISO/IEC 15504-1 standard [11] specifies an organizational unit (OU) as a part of an organization that deploys one process or has a coherent process context, and operates within a set of business goals and policies. An OU typically consists one part of a larger organization like one development team or regional unit, but small organizations may entirely exist as one OU. In other specifications which are based on ISO15504 [1], like TMMi2 [34], the relation between an organizational unit and rest of the organization is elaborated to allow overlying structures, like the upper management in the company, some steering activities, like policy control over the OU. However, the organizational unit remains a separate actor that operates by an internal process, being responsible for completing the task it has been assigned to, while complying with the policies set by upper organizations. The reason for using an OU as an assessment unit is that this way, the company size is normalized, making direct comparison between different types of companies possible.

Table 1. Interview rounds and themes

Round type	Number of interviews	Interviewee role	Description	Themes
1) Semi-structured	12 focus OU interviews	Designer or Programmer	The interviewee was responsible for or had influence on software design.	Design and development methods, Testing strategy and methods, Agile methods, Standards, Outsourcing, Perveiced quality
2) Structured with Semi-structured	31 OUs, including 12 focus OUs	Project- or Testing manager	The interviewee was responsible for the sofware project or testing phase of the software product.	Test processes and tools, Customer participation, Quality and Customer, Software Quality, Testing methods and -resources
3) Semi-structured	12 focus OU interviews	Tester or Programmer	The interviewee was a dedicated tester or was responsible for testing the software product.	Testing methods, Testing strategy and resources, Agile methods, Standards, Outsourcing, Test automation and services, Test tools, Perceived quality, Customer in testing

In this study, the population consisted of OUs from small, nationally operating companies to large internationally operating corporations, covering different types of software organizations from hardware producers to software houses and to contract testing and consulting services.

3.2 Data collection

The initial population and population criteria were decided based on prior research made by our research group [15-17, 32]. We carried out three interview rounds in our study (Table 1). The sample of the first and third interview round consisted of our focus group of 12 OUs collected from our research partners, and later supplemented by researchers to achieve a heterogeneous, polar type sample [5]. The second round of interviews was conducted as a survey with 31 OUs, including the focus group from the first round. Overall, the interviews were done during the winter of 2008-2009.

The 12 OUs in the focus group were professional software producers of a high technical level, with software development as their main activity. The selection of the focus group was based on the polar type selection [5] to cover different types of organizations. The focus group included different business domains and different sizes of companies. The organizations varied (Table 2) from software service consultants to software product developers, extending even to large hardware manufacturers, developing software for their own hardware products. The smallest OU in the focus group was a software developer with approximately twenty full-time employees; the largest was part of an internationally operating software producer employing over 10000 people.

The objective of this approach was to gain a broader understanding of the practice of and to identify general factors that affect test case selection and case prioritization. To achieve this, our research team developed two questionnaires and a survey that included questions on themes such as development methods, test processes, test phases, test tools, test automation and quality characteristics. The complete questionnaires and the survey form are available at http://www2.it.lut.fi/project/MASTO/. A reference list of the different themes in different data collection rounds is also available in Table 1.

The interviews contained semi-structured questions, and the whole sessions were tape-recorded for qualitative analysis and to further elaborate on different concepts during the latter rounds. Typically, an interview lasted for approximately one hour and they were arranged as face-to-face interviews with one organization participant and one or two researchers.

The decision to interview designers during the first round was

Table 2- Description of the Interviewed OUs

OU	Business	Company size[2] / Operation
Case A	MES[1] producer and electronics manufacturer	Small / National
Case B	Logistics software developer	Large / National
Case C	ICT consultant	Small / National
Case D	Internet service developer and consultant	Small / National
Case E	Naval software system developer	Medium / International
Case F	Safety and logistics system developer	Medium / National
Case G	Financial software developer	Large / National
Case H	ICT developer and consultant	Large / International
Case I	Financial software developer	Large / International
Case J	SME[2] business and agriculture ICT service provider	Small / National
Case K	MES[1] producer and logistics service systems provider	Medium / International
Case L	Modeling software developer	Large / International
19 survey-only cases	Varies; from software consultancies to software product developers and hardware manufacturers.	Varies

[1]Manufacturing Execution System [2]SME definition [6]

based on our aim to gain a better understanding of the operational level of software development. We wanted to see whether our hypotheses from our prior studies [15-17, 32] and literature review were valid. The interviewees in the first round were selected from a group of developers or programmers, who had the possibility to decide on or affect the structure of the software product. In one first-round interview, the organization interviewed was allowed to send two interviewees, as they considered that the desired role was a combination of two positions in their organization. In another first-round interview, we allowed the organization to supplement their answers, as the interviewee considered that the answers lacked some relevant details.

For the second round and the survey, the population was expanded by inserting additional OUs to enable statistical comparison of results. Selecting the sample was demanding because comparability was not specified by a company or an organization but by an OU with comparable processes. With the help of authorities (the network of the Technology and Innovation Agency of Finland) we collected a population of 85 companies. Only one OU from each company was accepted to the population to avoid bias of over-weighting large companies. From this list, the additional OUs accepted to the survey sample were selected according to the population criteria used in the first interview round.

We expanded the sample size in the second round to 31 OUs, including the OUs of the first round. The purpose of combining the interviews and the survey was to collect data more efficiently, simultaneously gaining a generalized perspective with survey-sized data, and obtaining detailed information about test management for the grounded analysis

During the second round of data collection, our decision was to interview and simultaneously conduct a survey where the population consisted of project or test managers. The objective was to collect quantitative data about the software and testing process and further to collect qualitative material about various testing topics, such as test case selection and agile methods in the software process. We selected managers for this round as they tend to have more experience about software projects; they have a better understanding of the overall software or testing process and the influence of upper management policies in the OU.

In the third round, the same sample organizations were interviewed as in the first round. The interviewees of the third round were testers, or in the case where the OU did not have separate testers, programmers whose tasks included module testing were interviewed. The interviews in these rounds focused on such topics as problems in testing (such as complexity of the systems, verification, and testability), the use of software components, testing resources, test automation, outsourcing, and customer influence in the test process.

The interview rounds, interviewee roles in the organization and study structure are summarized in Table 1, and the participating organizational units are summarized in Table 2.

3.3 Data Analysis

The grounded theory method contains three data analysis steps: open coding, where categories and their related codes are extracted from the data; axial coding, where connections between the categories and codes are identified; and selective coding, where the core category is identified and described [31].

The objective of the open coding was to classify the data into categories and identify leads in the data. The process started with "seed categories" [5] that contained essential stakeholders and known phenomena based on the literature. Seaman [29] notes that the initial set of codes (seed categories) comes from the goals of the study, the research questions, and predefined variables of interest. In our case, the seed categories were derived and further developed based on our prior studies on software testing, and from the literature. These seed categories were also used to define themes for the questions in the questionnaire, including topics such as development process, test processes, testing tools, automation or role of the customer. A complete list of the seed categories and general themes of the study is in Table 1.

In open coding, the classified observations are also organized into larger categories. New categories appear and are merged because of new information that surfaces during the coding. For example, our initial concept of having quality as a separate category was revised and quality was included within other categories such as criticality or outsourcing as an attribute with an "effect on quality". Another notable difference from the seed categories was that the management and policies were not as restrictive as originally thought, so they were incorporated into such themes as project management and test planning. Additionally, concepts like process difficulties or improvement proposals were given their own categories. At the end of the open coding, the number of codes was in total 166 codes, grouped into 12 categories.

The objective of the axial coding was to further develop separate categories by looking for causal conditions or any kinds of connections between the categories. In this phase, the categories and their related observations were becoming rigid, allowing the analysis to focus on developing the relationships between larger concepts. In this phase, the categories formed groups in the sense that similar observations were connected to each other. For example, codes such as "*Process problem: outsourcing*", "*Outsourcing: Effect to quality"* and "*Development process: support for outsourced activities*" formed a chain of evidence for observing how the outsourced resources in development fitted in to the overall process. By following this type of leads in the data, the categories were coded and given relationships with each other.

The third phase of grounded analysis, selective coding, was used to identify the core category [31] and relate it systematically to the other categories. As based on [31], the core category is sometimes one of the existing categories, and at other times no single category is broad or influential enough to cover the central phenomenon. In this study, the examination of the core category resulted in the category "applied test case selection approach", with a set of software testing concepts listing issues related to the core category or explaining the rationale for observed activities. The core category was formed by abstracting the categories and defining a common denominator, because none of the categories was considered influential enough to explain the entire phenomena. For example, we observed the primary case selection method in all of our organizations, but were unable to define one cause for the approach the organizations applied. Our initial approximation that the case selection method was closely connected to the development method and the role of the customer was partially correct, but we also identified several other aspects like amount of resources or test case developers, which also seemed relevant. Overall, we adjusted the core category to include all these concepts, which also became the categories presented in this paper. Additionally, by identifying the core category and affecting factors, we were able to define and name two approaches for selecting the approach for test case selection.

4. RESULTS AND OBSERVATIONS

In the following section we present and discuss the observations from our study. First of all, we were able to identify several concepts which would affect the case selection method, and introduce them in the first part. Secondly, we elaborated the observation made from the categories into hypotheses, which summarize and explain how organizations in this study selected test cases. Finally, in the third part we introduce our two stereotypes of selection methods.

4.1 Developed categories

The categories were developed based on their observed effect on actual test case selection and their ability to interpret why the organization had decided to use this approach. These categories were all related to the core category, identified during selective coding, and had a definite impact on how the organization approached test case selection or explained the differences between organizations. For example, the category *applied selection method* was taken directly from the observation data as it discussed the studied phenomenon of case selection, while *software type* and *development approach* were used to establish the objectives and operating methods of the software development organization. The category *selection problem* was also taken directly from observations as it discussed the difficulties in applying the used approach. The categories of *test designers*, *testing resources*, *customer influence* and *explorative testing* were included as they were observed to follow a pattern based on case selection method or otherwise clearly divided the respondents. The complete list and short summary of the developed categories are available in Table 3.

The category of *applied selection method* describes the primary way the organization selects what features or use cases are tested during development. Selection seems to be based on one of two major approaches: the risk-based “Which causes the largest expenses if it is broken?” and definition-based “Which are the main functionalities the software is supposed to do?”. In some organizations, there are also some secondary concerns like conformance testing to ensure that the system complies with some interface or with a set of established requirements, or “changes first”, where the most recent changes have priority over other test cases.

The category s*oftware type* defines the type of software the organization is building as their main product. In this study, the development outcomes were classified into three categories: s*oftware service, software product and software module for hardware*. In s*oftware service*, the software is used as a network-based application or front-end for network service, including Internet services. In s*oftware product*, the application is stand-alone software installed on a platform such as a PC or mobile phone. The last category, s*oftware module for hardware* refers to embedded software for dedicated devices.

The category *test designers* defines the personnel responsible for defining and designing test cases or authorized to decide on what area the testing effort is focused on. In several organizations, the test cases are designed by the programmers themselves or by designated software structure designers. The management level, made up of test managers or project managers, was responsible for designing test cases in five organizations, and the clients were allowed to define test cases in three organizations. Overall, the responsibility for test designing varied between organizations.

The category of *development approach* defines the approach the organization applies to software production. This category is defined based on a linear dimension defined by Boehm [2], where the polar points represent fully plan-driven and fully agile development, with overlap in the middle, combining techniques from both sides. For example, several organizations adopt only some activities from agile methods and apply them in the traditionally plan-driven environment, or apply the agile approach to smaller support projects, applying plan-driven methods to the main product development projects.

The category of *testing resources* is an indicator of how much resources the test organization has when compared to their optimal, i.e. perfect situation. In this category, we apply a scale with three possibilities, Low (33% or less), Moderate (34-66%) and High (67% or more). For example, if an organization currently has two dedicated testers and thinks that they could use three, it would mean a resource availability of 67 %, translating to “High” on the scale. It should be noted that in this scale, the score less than “High” does not necessary mean that the test process is inefficient; the scale is merely an indicator of the amount of resources allocated to testing tasks. The ratings, presented in the Table 4, are based on the answers given by the organization during the second round survey.

The category of *customer influence* defines the part customers have in the development process. The most common ways of influencing a project was by directly participating in some testing

Table 3. Categories defined in the empirical analysis

Category	Description
Applied selection approach	The method the organization is currently using to select which test cases are included in the test plan.
Software type	The type of software the OU is developing.
Test designers	The personnel responsible for designing and selecting the test cases.
Development approach	The method the organization is currently using to develop software.
Testing resources	An approximation on how large an amount of testing resources the organization currently has access to, in comparison to the optimal, ie. perfect amount of resources.
Customer influence	The type and method of customers to influence the organization’s software test process.
Selection problem	The most common process hindrance the test case selection method causes to the organization.
Explorative testing	Does the organization apply non-predefined test cases in their test plan?

phase, by approving the test results or approving the test plan made by the developer organization.

The category of *selection problem* defines the process hindrances caused by the test case selection approach. In risk-based selection, the common hindrances were that the test cases either did not cover all of the important cases or that the designed cases were discarded from the final test plan. With the design-based approach, the problems were usually at the management level, being caused by such concepts as restrictive test policies or managing test process to meet all required and formal activities, like communications, paperwork, schedules, test environments, weekly reviews, project steering group meetings and such; In layman's terms, the increased amount of red tape.

Finally, the category of *explorative testing* indicates whether or not the organization applies explorative testing methods. For this category, all testing methods which apply non-predefined test types, like interface or usability-testing, were considered explorative. In this category, the organizations were strongly divided into two opposing groups; some organizations considered explorative testing as an important phase where usability and user interface issues were addressed, whereas some organizations considered testing without test cases and documentation as a waste of test resources.

4.2 Hypotheses and Observations

Our study developed hypotheses based on the observations regarding test case selection. The hypotheses were shaped according to the categorized observations listed in Table 4, by developing concepts that explained the observations and followed the rational chain of evidence in the collected data. For example, the first hypothesis was generalized from the observation that all organizations that applied a design-based approach also favored plan-driven product development, and that their customers tended to have influence on the design-phase of the product. Following this lead, we focused on these observations and tried to define exactly how the risk-based approaches differed in the design-phase and would this observation be generalizable enough for creating a hypothesis. A similar approach was used with hypotheses two and three. The last hypothesis, number four, came from the general observation that for some reason, several organizations considered explorative testing to be wasteful or a futile waste of resources,

Table 4. Observations on test case selection method

Case	Applied selection method	Software type	Test designers	Development approach	Testing resources	Customer influence	Test case selection problem	Explorative testing
A	Risk-based with changes first	Software module for hardware	Programmers	Plan-driven supported by agile	Low	Approves product	Important test cases are discarded	Yes, programmers do it.
B	Risk-based	Software product	Designers	Agile	Moderate	Participates in testing	Agile products seem to be difficult to test.	No, only defined cases are tested.
C	Risk-based with changes first	Software product	Programmers with clients	Agile	Moderate	Participates in testing	Some test cases are not implemented.	Yes, programmers do it.
D	Risk-based	Software service	Programmers	Plan-driven supported by agile	Low	Approves testing plan	Some test cases are not implemented	Yes
E	Risk-based	Software module for hardware	Programmers	Agile supported by plan-driven	High	Approves product	Important test cases are discarded	Yes, some phases apply.
F	Risk-based with conformance	Software module for hardware	Designers	Plan-driven	Moderate	Approves product	Some test cases are not implemented	Yes
G	Design-based with conformance	Software service	Test manager with testers	Plan-driven	High	Approves testing plan	Validating functionalities is difficult.	No, only defined cases are tested.
H	Design-based	Software service	Designers with clients	Plan-driven	High	Approves testing plan	Amount of policies affect test effectiveness.	No, not enough time.
I	Design-based	Software service	Test manager with testers	Plan-driven	High	Approves design	Too large reliance on test manager experience	No
J	Risk-based, changes first	Software product	Project manager	Plan-driven supported by agile	High	Participates in testing	Important test cases are discarded	Yes
K	Design-based	Software module for hardware	Project manager, clients	Plan-driven supported by agile	Moderate	Participates in test design	Some test cases are not implemented	Yes, in some projects.
L	Design-based	Software product	Project manager with designers	Plan-driven	High	Approves product	Test management in large projects	Yes, several phases apply.

whereas some thought that it was one of the most important aspects in testing. As this behavior was not as systematic with our observations as some other aspects of test case selection, it was included as a separate observation, and subsequently a separate hypothesis, on test case design and case selection.

Hypothesis 1: Risk-based selection is applied when the software design is not fixed at the design phase. Risk-based selection was used in all those organizations that applied primarily agile development methods in their software process. Furthermore, all organizations that applied traditional plan-driven software development methods also applied the design-based test case selection approach. With the risk-based approach, the selection was clearly based on communication between case selectors and stakeholders:

"Basically our case selection method is quite reactive [to feedback]." -Case E, Tester

"I might use risk-based techniques based on the advice from developers." – Case B, Designer

In the design-based approach, software process management gets much more involved:

"Test manager decides based on the requirements on what will be tested." – Case G, Tester

"Designers with the project manager decide on the test cases." – Case L, Designer

In general, it also seemed that in the organizations that applied the risk-based approach, customers had a big influence on the latter parts of the software process, either by approving the final product or by directly participating in the latter test phases.

"…so far we have been able to go by trusting [final] testing phases to the customer." – Case C, Designer

"For larger projects we give our product to a larger client for test run and see how it works." – Case A, Tester

In organizations applying the design-based approach, the customer input in test design was more indirect, including approaches like offering supplemental test cases or reviewing case selections.

"…customers can come to give us their test case designs so we can accommodate to their requirements." –Case K, Designer

"Customer usually gives input on test design if the test plan has shortcomings or is overly vague." – Case H, Tester

Hypothesis 2: The design-based approach is favored in organizations with ample resources and but it requires more management. The most definite difference between organizations that chose the design-based approach was that most of them reported a high amount of testing resources. On average, companies with the design-based approach had 73% of required resources, while in the risk-based group, the average was 49%.

Another indicator of the differences between the two groups was the types of problems the testing process experienced in their case selection. In the risk-based selection, the most common process difficulty was related to the test cases. Either they did not cover all the critical cases, or they discarded critical cases from the final test plan.

"The problem is in defining what should be tested." –Case A, Designer

"The document quality fluctuates between projects… sometimes the critical test cases should be defined more clearly." – Case C, Designer

"What we truly miss is the ability to test all modules consistently." – Case D, Designer

In the design-based approach, the most common problems were related to managing the testing process, satisfying testing criteria set by test policies or keeping up with the requirements.

"It is up to test managers and their insight to define a satisfying test case design." – Case I, Designer

"We are already at the full capacity with test cases, we should start discarding some of them..." – Case K, Tester

"[Policy makers] really cannot put testing into a realistic schedule." – Case H, Tester

An interesting observation was that in the design-based approach, the test cases were mostly designed by a separate test process management or test managers, whereas in the risk-based approach the design was done by software developers: programmers or software designers.

Hypothesis 3: The use of test automation is not affected by the case design or case selection approach. The effect of test case selection approaches on feasibility or applicability of test automation was also examined, but the results did not yield any relevant information or distinct pattern. Aside from prior observations on test automation [15], the decision of applying test automation did not seem to be connected to test case selection, meaning that the decision to implement automation is based on other test process factors. For example, Case B from the risk-based group was an active user of test automation services:

"All our projects have test automation at one time or another." – Case B, Designer

In the design-based approach, Cases G and K had a significant number of automated test cases in their software development process:

"…for some ten years most of our conformance test cases have been automated." – Case G, Tester

"Well, we have test cases which are automatically tested during the nighttime for every daily build." –Case K, Tester

In fact, all of the organizations had some forms of automation, were introducing automation to their test process or could see some viable way of applying test automation.

"Regression tests which are build on our own macro-language, and some unit tests." – Case G, Designer

"[new testing tool] is going to allow automation." – Case A, Manager

"We are implementing one for interface testing" – Case I, Tester

Hypothesis 4: Explorative testing may be seen by policy makers as an unproductive task because of its ad hoc nature. The explorative testing methods in this study included all test methods and practices where testers did non-predefined test activities as a part of the standard test process. In organizations where the risk-based approach was applied, explorative testing was commonly applied, whereas in the design-based approach the amount of exploration was noticeably smaller.

"[programmers] are allowed to do tests as they please" –Case A, Designer

"Yes we do that, however the benefit of that work varies greatly between individuals." –Case E, Tester

"Those 'dumb tests' really bring up issues that escaped developers' designs." – Case F, Tester

However, by comparing the organizations based on their originating company sizes, it becomes evident that large-sized companies used less explorative test methods. One reason for this could be that explorative testing is difficult to document; in other words the explorative test process would cause additional requirements for management and policies.

"We have so much other things to do…no time for that [explorative testing]" – Case H, Tester

"It would be interesting but no, we do not do that kind of thing." – Case I, Tester

"Well maybe if there were some unusual circumstances but I think no; even in that case we would probably first make plans." – Case G, Tester

4.3 Observed approaches

Based on the observations above, we are able to conclude that the software case selection approaches tend to resemble two basic approaches; risk-based and design-based selection. Typically in the risk-based approach, test design tasks are planned and completed by software developers, whereas in the design-based approach, management and separate test managers are responsible for the case generation. It seemed that the organizations applying risk-based approaches were also more likely to apply agile methods in their software processes. However, also some design-based organizations applied agile methods if it was deemed necessary. This behavior could be explained with customer participation. As the risk-based approach also favors customer participation in the latter parts of the process, it allows a customer to request last-minute changes. As agile development does not create a strong, "iron-bound" [2] design for the software product, but rather general guidelines for development objectives, it would also seem reasonable to assume that test cases are selected based on foreseeable risks and not based on the design documentation which may lack details. These two approaches are summarized in Table 5.

The selection of the risk-based approach was also favored when the testing resources were limited. If testing is organized with limited resources, the prioritization of test cases takes place, favoring the risk-based approach. In this situation, the obvious choice is to allocate resources to address the most costly errors. There are studies showing that by prioritizing test cases, the test process can be organized effectively with as low as 15% of the desired resources [25]. The costs caused by product defects offer an easy and straightforward measurement method to determine which cases should be tested and which discarded as an acceptable expense.

Besides serving as the method of test case prioritization, the risk-based approach was also more likely to supplement test cases with exploratory testing practices, a phenomenon that may be related to the test policy issues of the design-based approach. Where the design-based approach was applied, the organizations emphasized management and policies. The actual type of software product seemed to have little to no impact on selection approach.

5. DISCUSSION

As software testing usually only has a limited amount of resources [18], there always has to be some form of selection process on which parts of the software should be tested and which can be left as they are. Petschenik [25] discusses this phenomenon by implying that the testing process can be organized effectively with merely 15% of the required resources; Huang and Boehm [9] indicated that a 20% investment can cover 80% of the testing process if the test case design and test focus is selected correctly. In practice, we observed the same phenomena, as several organizations reported a resource availability of 60-70%, indicating that they do prioritization with their test cases.

Our study examined how test cases were designed and selected for test plans in 12 professional software organizations. The results indicate that test case selection seems to generalize into two approaches; risk-based and design-based. In the risk-based approach, test cases are selected on the basis that most costly errors are eliminated from the software. In many cases it is the economically preferable strategy to keep deadlines rather than to extend testing phases [9,27]. In these cases, testing resources are more likely geared towards minimizing the costs caused by errors found after the release.

Table 5. Two stereotypical approaches for test case selection

Category	Risk-based selection	Design-based selection
Test designers	Developers: programmers and testers	Managers: test and project managers
Development approach	Leans towards agile methods	Leans towards plan-driven methods
Testing resources	Limited	Sufficient
Explorative testing	Applied commonly	Applied rarely
Effect of policies in decisions on testing.	Small; most decisions done in project level.	Large; most decisions are based on company policies or customer requirements.
Customer influence	In the testing process	In the design process
Limitations of the model	Test case coverage may become limited.	Test process may become laborous to manage
Design concept	"What should be tested to ensure smallest losses if the product is faulty?"	"What should be tested to ensure that the product does what it is intended to do?"

The other selection method is the design-based approach. In this approach, the organization decides the test cases based on the design documentation of the product, ensuring that the software is capable of performing the tasks it is supposed to do. The design-based approach seems to be favored in organizations that have sufficient or ample testing resources. These organizations may also have stricter customer-based or policy-defined activities in their software process, like following a strict formal process, or requiring customers to approve all decisions and expenses related to the project. The most common process hindrances in the design-based approach seem to be policy restrictions and management issues like rigid processes, top-heavy management and communicating between all relevant stakeholders. As for selection between the two approaches, a crude definition can be drawn based on process stability. If the development process is predictable and process outcomes are detailed, then the design-based approach is mostly feasible. If the process more likely responds to changes during the development, then the risk-based approach is preferred.

Obviously, a limitation of this study is the number of organizations. Our study interviewed 36 software professionals from 12 different organizations, which were selected to represent different types and sizes of software organizations. For this type of study, Onwuegbuzie and Leech [37] discuss the several threats associated to the validity. In their opinion, internal validity and external credibility should be maintained by providing enough documentation, explaining the applied research method and providing proof of the chain of evidence that led to the study results. In this project, the internal validity was maintained with these viewpoints in mind. For example, we applied several researchers in designing the questionnaires, and later the same researchers collected, and subsequently analyzed the data. In addition, we conducted a survey in 31 organizations to collect quantitative data to compare and cross-reference our qualitative observations with quantitative data.

The objective of this qualitative study was not to establish statistical relevance, but to observe and explain the strategies of how real-life organizations decide which test cases to select. Our analysis revealed two selection approaches with several characterizing attributes explaining the differences. However, they also shared some attributes like software types, so in practice they more likely complement each other and the division is not as straightforward as it may seem based on the results.

6. CONCLUSIONS

In the observed organizations, test cases were selected using two main approaches: the risk-based and the design-based approach. Generally, in organizations where testing resources were limited and the product design was allowed to adapt or change during the process, the risk-based approach became increasingly favored. When the project was allowed more testing resources and the software design was made in a plan-driven fashion, the objective for the test process shifted towards test case coverage, and subsequently, towards the design-based approach in test case selection. In these cases, the case selection was based on product design and the verification of features, not in damage prevention and minimizing the possible risks. However, in practice the shift between approaches was not as clear-cut as it may seem; additional concepts like policies, customers and development methods can also affect the selection.

We observed and presented results on how software test cases are selected and how test plans are constructed with different amounts of resources in different types of software organizations. We believe that software organizations can achieve better productivity by defining the test process and by focusing on the critical aspects for test process. By designing the test cases to more closely to fit the needs of the organization and product characteristics, test process issues can be better addressed and more attention can be given to the aspects that need enhancement. Therefore, these results can be used to develop testing practices and generally to promote the importance of designing test plans to fit the process organization.

7. ACKNOWLEDGMENTS

This study was supported by the ESPA project (http://www.soberit.hut.fi/espa/), funded by the Finnish Funding Agency for Technology and Innovation, and by the companies mentioned at the project web site.

8. REFERENCES

[1] Bertolino, A., “The (Im)maturity Level of Software Testing”, ACM SIGSOFT Software Engineering Notes, Vol. 29(5), 2004, pp. 1-4, DOI: 10.1145/1022494.1022540

[2] Boehm, B., “Get Ready for the Agile Methods, with Care”, Computer, Vol. 35(1), 2002, pp. 64-69, DOI: 10.1109/2.976920

[3] Chen, Y. Probert, R.L. and Sims, D.P., “Specification-based Regression Test Selection with Risk Analysis”, Proc. 2002 conference of the Centre for Advanced Studies on Collaborative research, 30.9.-03.10., Toronto, Ontario, Canada, 2002.

[4] Do, H. and Rothermel, G., “An Empirical Study of Regression Testing Techniques Incorporating Context and Lifetime Factors and Improved Cost-Benefit Models”, Proc. 14th ACM SIGSOFT international symposium on Foundations of software engineering, 5-11.11., Portland, Oregon, USA, 2006, pp. 141-151. DOI: 10.1145/1181775.1181793

[5] Eisenhardt, K.M., "Building theories from case study research”, Academy of Management Review, Vol. 14, pp. 532-550, 1989.

[6] EU, "SME Definition," European Commission, 2003.

[7] Gill, N.S., “Factors Affecting Effective Software Quality Management Rivisited”, ACM SIGSOFT Software Engineering Notes, Vol. 30(2), 2005, pp. 1-4. DOI: 10.1145/1050849.1050862

[8] Glaser, B. and Strauss, A.L., The Discovery of Grounded Theory: Strategies for Qualitative Research. Chicago: Aldine, 1967.

[9] Huang, L. and Boehm, B., “How Much Software Quality Investment Is Enough: A Value-Based Approach”, IEEE Software, Vol. 23(5), 2006, pp. 88-95, DOI: 10.1109/MS.2006.127

[10] Hulkko, H. and Abrahamsson, P., “A Multiple Case Study on the Impact of Pair Programming on Product Quality”, Proc. 27th international conference on Software engineering, 15.-

21.5., St. Louis, MO, USA, 2005, pp. 495-504, DOI: 10.1145/1062455.1062545

[11] ISO/IEC, ISO/IEC 15504-1, Information Technology - Process Assessment - Part 1: Concepts and Vocabulary, 2002.

[12] ISO/IEC, ISO/IEC 25010-2, Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) Quality Model, 2008.

[13] ISO/IEC, ISO/IEC 29119-2, Software Testing Standard – Activity Descriptions for Test Process Diagram, 2008.

[14] Kaner, C., Falk, J. and Nguyen, H.Q., Testing Computer Software, 2nd edition, John Wiley & Sons, Inc., New York, USA, 1999.

[15] Karhu, K., Repo, T., Taipale, O. and Smolander, K., “Empirical Observation on Software Test Automation”, Proc. 2nd International Conference on Software Testing, Verification and Validation (ICST), 1-4.4., Denver, Colorado, USA, 2009.

[16] Karhu, K., Taipale, O. and Smolander, K., “Outsourcing and Knowledge Management in Software Testing”, Proc. 11th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2-3.04., Staffordshire, England, 2007.

[17] Kasurinen, J., Taipale, O. and Smolander, K., “Analysis of Problems in Testing Practices”, proc. 16th Asia-Pacific Software Engineering Conference (APSEC), 1-3.12., Penang, Malaysia, 2009.

[18] Kit, E., “Software Testing in the Real World: Improving the Process”, Addison-Wesley, Reading, MA, USA, 1995.

[19] Klein, H.K. and Myers, M.D., "A set of principles for conducting and evaluating interpretive field studies in information systems”, MIS Quarterly, Vol. 23, pp. 67-94, 1999.

[20] Kolb, R. and Muthig, D., “Making Testing Product Lines More Efficient by Improving the Testability of Product Line Architectures”, Proc. ISSTA 2006 workshop on Role of software architecture for testing and analysis, 17-20.7., Portland, Maine, USA, 2006, pp. 22-27, DOI: 10.1145/1147249.1147252

[21] Mao, C., Lu, Y. and Zhang, J., “Regression Testing for Component-based Software via Built-in Test Design”, Proc. 2007 ACM Symposium on Applied Computing, 11-15.3., Seoul, South Korea, pp. 1416-1421. DOI: 10.1145/1244002.1244307

[22] Meyer, B., “Design and code reviews in the age of the Internet”, Communications of the ACM, Vol. 51(9), 2008, pp. 66-71.

[23] Nørgaard, M. and Hornbæk, K., “What Do Usability Evaluators Do in Practice? An Explorative Study of Think-Aloud Testing”, Proc. 6th Conference on Designing Interactive Systems, 26-28.6., University Park, PA, USA, 2006, pp. 209-218, DOI: 10.1145/1142405.1142439

[24] Pare´, G. and Elam, J.J., “Using case study research to build theories of IT Implementation”, IFIP TC8 WG International Conference on Information Systems and Qualitative Research, Philadelphia, USA, 1997.

[25] Petschenik, N.H., “Practical Priorities in System Testing”, IEEE Software, Vol. 2(5), 1985, pp. 18-23, DOI: 10.1109/MS.1985.231755

[26] Redmill, F., “Exploring risk-based testing and its implications”, Software Testing, Verification and Reliability, Vol. 14(1), 2004, pp. 3-15, DOI: 10.1002/stvr.288

[27] Rosas-Vega, R. and Vokurka, R.J., “New product introduction delays in the computer industry”, Industrial Management & Data Systems, Vol. 100 (4), 2000, pp. 157-163.

[28] Rothermel, G., Elbaum, S., Malishevsky, A.G., Kallakuri, P. and Qiu, X., “On Test Suite Composition and Cost-Effective Regression Testing”, ACM Transactions on Software Engineering and Methodology, Vol. 13(3), 2004, pp. 277-331. DOI: 10.1145/1027092.1027093

[29] Seaman, C.B., "Qualitative methods in empirical studies of software engineering", IEEE Transactions on Software Engineering, Vol. 25, pp. 557-572, 1999.

[30] Shi, Q., “A Field Study of the Relationship and Communication between Chinese Evaluators and Users in Thinking Aloud Usability Tests”, Proc.5th Nordic conference on Human-computer interaction: building bridges, 20-22.10., Lund, Sweden, 2008, pp. 344-352, DOI: 10.1145/1463160.1463198

[31] Strauss, A. and Corbin, J., Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Newbury Park, CA: SAGE Publications, 1990.

[32] Taipale, O., and Smolander, K., “Improving Software Testing by Observing Practice”, Proc. 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE), 21-22.9., Rio de Janeiro, Brazil, 2006, pp. 262-271.

[33] Tassey, G., “The Economic impacts of inadequate infrastructure for software testing”, U.S. National Institute of Standards and Technology report, RTI Project Number 7007.011, 2002.

[34] TMMi Foundation, Test Maturity Model integration (TMMi) reference model, Version 2.0, 2009.

[35] Voas, J., Payne, J., Mills, R. and McManus, J., “Software Testability, An Experiment in Measuring Simulation Reusability”, ACM SIGSOFT Software Engineering Notes, Vol. 20, Issue SI, 1995, pp. 247-255. DOI: 10.1145/223427.211854

[36] Whittager, J.A., “What is Software Testing? And Why Is It So Hard?”, IEEE Software, Vol. 17(1), 2000, pp.70-79, DOI: 0.1109/52.819971

[37] Onwuegbuzie, A.J. and Leech, N.L., “Validity and Qualitative Research: An Oxymoron?”, Quality and Quantity, Vol. 41(2), April 2007, pp. 233-249. DOI: 10.1007/s11135-006-9000-3

[38] Yoo, S. and Harman, M., “Pareto Efficient Multi-Objective Test Case Selection”, Proc. 2007 international symposium on Software testing and analysis, 9-12.7., London, England, pp. 140-150. DOI: 10.1145/1273463.1273483

Publication V

How Test Organizations Adopt New Testing Practices and Methods?

Kasurinen, J., Taipale, O. and Smolander, K. (2011), Proceedings of the Testing: Academic & Industrial Conference: Practice and Research Techniques 2011 (TAIC PART) co-located with 4th IEEE International Conference on Software Testing, Verification and Validation (ICST), 25.3.2011, Berlin, Germany, doi: 10.1109/ICSTW.2011.63

How Test Organizations Adopt New Testing Practices and Methods?

Jussi Kasurinen, Ossi Taipale and Kari Smolander
Software Engineering Laboratory
Department of Information Technology
Lappeenranta University of Technology
Lappeenranta, Finland
jussi.kasurinen | ossi.taipale | kari.smolander@lut.fi

Abstract— **Software testing process is an activity, in which the software is verified to comply with the requirements and validated to operate as intended. As software development adopts new development methods, this means also that the test processes need to be changed. In this qualitative study, we observe ten software organizations to understand how organizations develop their test processes and how they adopt new test methods. Based on our observations, organizations do only sporadic test process development, and are conservative when adopting new ideas or testing methods. Organizations need to have a clear concept of what to develop and how to implement the needed changes before they commit to process development.**

Keywords-test process improvement; adoption of test methods; qualitative study; test process standard

I. INTRODUCTION

Software testing is an activity, in which the software product is verified to comply with the system requirements and validated to operate as intended [1]. In spite of this quite clear definition, testing cannot exist as a static process, which is separated from other activities of software development. There are several considerations on how testing should be done. For example, there exist different techniques like usability testing or test automation, which both require different testing tools and enable the test process to find different kinds of errors. Also several other factors such as customer participation, quality requirements or upper management affect the testing work [2, 3].

In this study, we observe ten software development organizations and their test organizations, representing different types of organizations that do software testing. Our purpose is to understand how these organizations manage and develop their test processes and adopt new ideas to their existing testing methods. Our focus will be on two aspects: on the ability to adopt new testing methods to the existing test process and on the ability to develop the test process itself to a desired direction. As a part of the latter aspect, we also conducted a feasibility study on the proposed test process model presented in the ISO/IEC 29119 software testing standard working draft [4]. Overall, the main research questions were "How organizations adopt new ideas to their test processes?" and "How feasible does the standard test process model ISO/IEC 29119 seem in practice?"

This paper continues our studies of software testing organizations [5,6]. The study elaborates on the previous studies by observing the test process itself, separated from the practical effects of different testing-related aspects such as testing tools and automation, test case selection method or development process, studied in the prior publications.

The rest of the paper is constructed as follows: Section 2 discusses the related research topics and introduces the standard process model used in the study. Section 3 introduces the applied research approach and Section 4 shows the findings of the study, which are then discussed and analyzed further in Section 5. Finally, in Section 6 the paper is wrapped up with conclusions.

II. RELATED RESEARCH

Testing strategy has been defined as a concept in several industry standards or certification models [for example 4, 7]. In the draft of the upcoming software testing process standard ISO/IEC 29119 [4] the test process is composed of several layers. The top layer in this model is the organizational test process level (Figure 1), which defines the testing policy and the testing strategy of the entire organization. The second layer is the test management process level, which defines the test activities in projects. On this level, test plans are defined and maintained based on the given organization level policies and strategies. The last level is the test processes level, which defines the actual testing work. This reference model is not by any means the first or only attempt to build a model for test processes. For example, the TMMi [7] framework defines a maturity-based assessment model for software testing. However, as TMMi is a maturity model, it is geared towards identification of process problems and improvement objectives, whereas ISO 29119 is aimed to provide an abstract model for good testing practices.

The software process improvement (SPI) literature includes studies about the effect of different factors in process improvement. For example, a study by Abrahamsson [2] discusses the requirements for successful process improvements. The most important factor according to this study is the commitment to change at all organizational

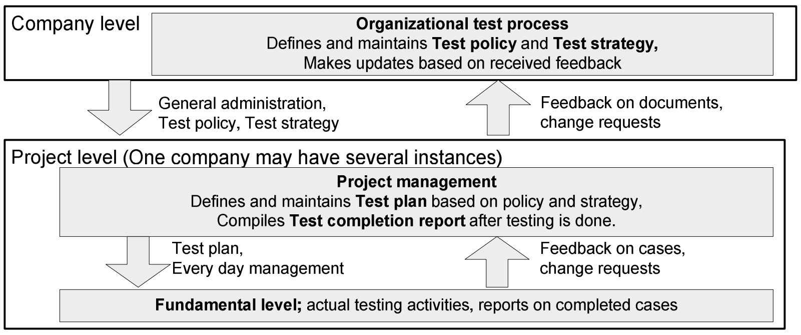

Company level
Organizational test process
Defines and maintains Test policy and Test strategy,
Makes updates based on received feedback
General administration,
Test policy, Test strategy
Feedback on documents,
change requests
Project level (One company may have several instances)
Project management
Defines and maintains Test plan based on policy and strategy,
Compiles Test completion report after testing is done.
Test plan,
Every day management
Feedback on cases,
change requests
Fundamental level; actual testing activities, reports on completed cases

Figure 1. ISO/IEC 29119 Standard test process model in a nutshell.

levels. If some of the levels disagree with the process improvement, SPI tends to fail.

In studies applying certain process models in organizations, Hardgrave and Armstrong [8] observed that their case organization had trouble reflecting their existing processes to given models. The organization estimated the time needed for process improvements to 10 months, when in fact the entire process development took four years. Hardgrave and Armstrong also concluded that organizations tend to lose the initial drive for process improvement because in many cases the internal need to develop is not the driver for improvement - instead improvement is seen as a means to reach out for certain external rewards, like certifications.

Dybå [3] conducted a study on SPI activities in different types of organizations. Dybå concluded that the company size does not hinder or restrict the process improvement activities. Small organizations are at least as effective as large ones in implementing process improvement, as they tend to be less formal in organizational hierarchy and to use explorative methods more willingly. Another observation was also that organizations have a tendency to define their own best practice methods, as in what is working, while failure in process improvement is considered unacceptable possibility. As process improvement projects often fail, companies tend to support status quo if corrective actions are not absolutely necessary.

III. RESEARCH METHOD

As a qualitative study, the selection of the study organizations was crucial to ensure that they minimized a possible result bias caused by too homogeneous study population. Our decision was to observe a heterogeneous group of organizations, with minimal bias caused by the application area or used software development methods. Based on these preferences, we selected ten organizations from our industrial collaborators and contacts to represent different types of organization sizes [9] and operating domains. Organizations included sizes from small to large, international and national businesses, from professional testing experts to service developers and organizations testing embedded software platforms. In addition, all our organizations were selected on the criteria that they tested software professionally and as a part of their main business activity. The list of the participating organizations is in Table 1.

We used organizational unit (OU) as our unit of analysis [10]. An organizational unit has at least one internal process, or activity which it conducts independently, receiving only guidelines and overview from the corporate level management above it. In large organizations, a unit like a department or a local office may constitute one OU, but in micro and small-sized companies, an OU may include the entire company. This way the size difference of case

TABLE I. DESCRIPTION OF THE OBSERVED OUS.

OU	Business domain, product type	Company size[2] / Operation
Case A	ICT developer and consultant, service producer	Small / National
Case B	Safety and logistics systems developer, software products	Medium / National
Case C	Financial and logistics software developer, software products	Medium/ National
Case D	MES[1] producer and logistics system provider, embedded software for hardware products	Medium / International
Case E	MES[1] producer and electronics manufacturer, embedded software for hardware products	Small / National
Case F	Maritime software systems developer, software products	Medium / International
Case G	ICT consultant specialicing in testing, test consulting services	Medium / National
Case H	Modeling software developer, software products	Large / International
Case I	ICT developer and consultant, software production consulting	Large / International
Case J	ICT consultant specialicing in testing, test consulting services	Small / National

[1]Manufacturing Execution System

organizations was normalized.

A. *Grounded Theory Approach*

Our study was an interpretative qualitative study, with main data collection method being interviews with case organization representatives. In data analysis, our team applied the grounded theory approach [11-13]. The original grounded theory method was defined by Glaser and Strauss [11], and was later elaborated into two similar, but different approaches. The Glaserian [13] approach is fundamentally founded on non-intrusive observation and emergence, while the Strauss-Corbin [12] relies on systematic codification and categorization process for observations. Because of a relatively large number of organizations for the qualitative study and practical difficulties on arranging a non-intrusive observation possibilities, we decided on applying the Strauss-Corbin approach.

The Strauss-Corbin-based grounded theory includes three steps for data analysis. The first step is called open coding, in which the collected data is codified to conceptual codes and grouped into higher level categories. The categories are created during the coding or some of them may be derived from, for example, seed categories [14], interview themes or research questions. Overall, during open coding the categories are separated, joined, created and deleted to understand and explain the data from the viewpoint of the research questions.

The next step is called axial coding. It can be started after the categories and observations have become somewhat stable. In this phase, the connections between different categories are explored and a conceptual mapping is done to establish connections between them.

The last step is the selective coding, in which the core category is established. The core category is the central phenomenon or activity, which is related to most if not all observed activities. The core category can be one of the existing categories, or an abstract class combining the other categories. After the core category is identified, the categorized findings are refined to form hypotheses, which summarize the observed activities, and further elaborated to a grounded theory model. In this study, we decided the core category to be *Management of Test Process Development*.

B. ***Data Collection***

The data for the grounded analysis was collected by approximately one hour long interviews with a semi-structured list of questions. For each interview, the participating organization selected one representative whom they considered most suitable for the interview. Our preference, and the most usual interviewee, was a project management level interviewee, like a test manager or project leader. When an interview was agreed on, the interviewee was given a compiled interview material, which contained short description of the ISO/IEC 29119 test process model, the list of terminology applied in the standard, a brief descriptions of the other interview topics, and a questionnaire form, which contained all the formal, structured questions of the interview.

The interviews were conducted by the researchers to ensure that the interviewees understood the questions similarly, and tape-recorded for later transcription and analysis. In two organizations, two people were interviewed, as the organization considered this to be their best option. Also in one case, the interview was cancelled because of personal reasons, but in this case we accepted written responses submitted via email instead.

The interview themes were designed by three researches from our research group, and tested for feedback with colleagues who had previous experience on conducting software engineering studies. Before the actual data collection interviews, the questions were also tested with an test interview on a pilot company that otherwise did not participate on the study. Final versions of the questionnaire and introductory material for the interviews are available at the address *http://www2.it.lut.fi/project/MASTO/*.

IV. RESULTS

The results are divided into two parts; in the first part we present the categories that we observed to influence the test process development and introduce the results from the feasibility assessment of the ISO/IEC 29119 model [4]. In the second part, we present and discuss the rationale behind the generalized model of how test organizations adopt new practices.

A. *Categories and Observations*

We derived the seed categories to the analysis from the results and observations of our previous studies [7, 8]. Our objective in the analysis was to find answer for the following questions: "How organizations adopt new ideas to their test processes?" and "How feasible does the standard test process model ISO/IEC 29119 seem in practice?" By analyzing the data, we formulated five categories that explained the test process improvement process and feasibility of the standard model. We also made several observations that allowed us to further elaborate the collected data to five major observations, and generated a model that provided an explanation of how the organizations developed their test process and adopt new test practices. These observations and their respective categories are listed in Table 2, and the model is presented in Figure 2.

The first category is the *test documentation*, which depicts how the existing testing process is documented in the current organization. The category test documentation also contains information on how much detail and which kind of information concerning the testing work currently exists in the organization.

The second category is the *test process development*. This category is used to describe how often, and with what kind of activities the organization develops its test processes.

The third category is the *adoption of new methods*. This category explains how the organization adopts new test methods, on which they have no existing knowledge or experience. The category covers the concept of learning about a new test method without hands-on experience, and the willingness of allocating resources to test it in practice.

TABLE II. OBSERVATIONS IN TEST PROCESS DEVELOPMENT.

	Test documentation	Test process development	Adoption of new methods	Usage of experience and feedback	Applicability of the standard model
Case A	Quality system defines software process, guidelines for testing exist.	Constantly developed and maintained, part of quality system.	Evaluation, not necessarily tried out in practice.	Sometimes, used to develop test suite.	Seems usable; not taking into account the customer - weakness.
Case B	Quality system defines software process, guidelines for testing exist.	Documents, process updated if needed.	Would try, but not actively looking new methods.	Sometimes, little actual effect.	Seems usable.
Case C	Informal, unwritten policies. Guidelines agreed within group.	Trial and error, stick with what seems to be working, discussed if needed.	Would try, but not actively looking new methods.	Always, learning from errors promoted.	Not usable; too much documentation. Seems straightforward to implement, good amount of abstraction.
Case D	Test documentation exists, lacks details.	Documents, process updated if needed.	Would try, sometimes actively tries new methods.	Always, previous knowledge used in continuance projects.	Usable; more details in high level than needed.
Case E	Informal, unwritten policies. Guidelines agreed within group.	Guidelines updated if needed, no written documentation.	Without any previous knowledge, no.	Rarely, comparing between projects is considered unfeasible.	Seems usable, could use a list of how important different modules are.
Case F	Quality system defines software process, guidelines for testing exist.	Process updated regularly, discussions, sometimes changes are reverted.	May be piloted, and then decided if taken into use.	Almost always, little actual effect.	Seems usable; too much details in high levels, good reference for names and terms.
Case G	Test documentation exists, is tailored to suit projects.	Documents, process tailored per project from generic model.	Would try, central part of business.	Always.	Usable, not many novel concepts.
Case H	Test documentation exists, lacks details.	Documents, process updated if needed.	Without any previous knowledge, no.	Always, some actual effect.	Seems usable, more details in high level than needed.
Case I	Test documentation exists, is tailored to suit projects.	Process evaluated after every project.	Evaluation, not necessarily tried out in practice.	Always, some actual effect,	Seems usable; needs more scalability; too much documentation.
Case J	Test documentation exists, is tailored to suit projects.	Updates if needed, systematic overview once every few years.	May be piloted, depends on source credibility.	Always, learning from errors promoted.	Seems usable, needs more scalability.

The fourth category, *use of experience and feedback*, describes how the organizations use their previous experiences in test process development. This category is based on the concept on ISO/IEC 29119 [6] standard process model, in which every test process level creates feedback for upper level management.

The fifth category is the *applicability of the standard model*. In this category, the summary of the feedback about their opinions on the ISO/IEC 29119 standard process model is presented.

Based on the categorized data and observations made from the interview data, following five observations were made:

1) All organizations had defined roles for test plan development.

In every case organization, the test plan was designed by one dedicated person, holding the role accountable for the task. Usually this person was either test manager or tester with most suitable experience. However, the maturity of the plan varied; in cases C and E the test plan was merely an agreement over focus areas and priorities, while Case G made detailed, tailored documentation for testers to follow.

2) Test documentation seems to be feasible to implement as defined in the standard model.

In all case organizations, the test documentation defined in the ISO/IEC 29119 standard, test policy, test plan, test strategy and test completion reports, were considered feasible. In theory, all organizations agreed that they would be able to define these documents based on their current organization. In fact, in several cases the documents already existed. However, the practical implementation varied, in some organizations they were a part of quality system, and in some, unofficially agreed guidelines on testing work. For application of test completion reports, the problem was usually in the use of the completion reports in review and follow-up phases. Even if the documents existed, there were reported cases where the feedback in the test completion report was not really used, or that projects did not always bother to collect feedback and keep post mortem meetings.

"All projects should have post mortems, but all projects don't have post mortems. So that's again, the written, process description versus real life." –Case F

3) Project level application of the test process is usually more in line with the standard model than management.

This observation was based on the feedback from the ISO/IEC 29119 standard model and on the comments made in the interviews. In several organizations, the existing project level activities were very similar to the standard model, but the high-level management was considered unnecessarily detailed or too complex. In three case organizations, cases D, F and H, this was most obvious and in fact addressed as a concern over the standard model.

"I would say, that it suits for us quite well. Of course we don't have the upper level so much detailed, it is just… the common understanding about [how management works]" – Case D

4) Using feedback to systematically develop the test process is usually the part missing.

In most organizations, the common way to develop the test process was to implement changes "if needed". This was the mindset in six case organizations. This combined with the observation that test completion reports were used in several of those cases (C, H and J) would indicate that the feedback from the test completion reports was not systematically used.

"We have a meeting after the project where we also consider how the testing has been successful, how it has been done. And we try to learn from these meetings. Sometimes we get new good ideas from those, but not always." –Case H

In some organizations (cases A, F and I) the test process development was continuous, but even in those, the feedback from project-level to organizational level was usually missing. In Case A, the feedback was limited to develop the test tools, but did not affect the entire test process. In Case F, the feedback was used but the actual changes were minimal, and in Case I the test completion reports were sometimes skipped.

"[Do these reports affect how testing is done in later projects?]" "To be honest, I don't think so." –Case F

5) Organizations do not generally apply new ideas or try testing methods unless they have strong positive incentives for doing so.

The organizations were asked to evaluate, what would they do, if someone in the organization found out about new testing practice that would seem to offer improvements, but on which they had no previous experience or knowledge. Based on the responses, only two organizations (D and G) said that they would probably try it in an actual development project. Two other organizations considered that they would try it in a smaller pilot project (cases F and J). Two considered testing the method but also told that they are not looking for new methods or improvements (cases B and C).

"We're not bleeding edge people to try new, brand new testing practices. If we hear from many sources that it would be nice and interesting, then we might take a look." –Case C

Two organizations (A and I) said that they would evaluate the method on how it would theoretically fit to the organization, but not necessarily try it out. Last two organizations (E and H) considered that they would not be interested in a method without any first-hand knowledge or experience.

"Without prior knowledge, no."…"Because we don't have time to test too many new techniques, so we have to be quite sure in the beginning that it's worth testing, or the time." – Case H

B. How are new practices adopted?

Based on these observations and findings it seems plausible that organizations develop their process only when a clear need arises and do not tend to spontaneously try out new testing practices. If existing process works acceptably, the feedback from completed projects is ignored. From the viewpoint of the ISO/IEC 29119 standard model, the biggest differences seem to be in organizational management. The testing work in project-level is usually at least somewhat similar to the standard, but on the organizational level, the continuous development and feedback processes are usually missing. Many of the observations in process development and adoption of new testing practices seem to be related to the management decisions, whether in allowing resources to try out new concepts or willingness to implement changes. It also has large influence on what are the objectives of process development [2, 3]. Therefore the category *Management of test process development* can be considered the core category in this study, as it explains all the categories and has a relation to all observations.

If the observations are to be generalized to a grounded theory, it would seem that development happens only when the existing process obviously has a need to develop, and required resources for development are justified by the possible savings later. The existing test process becomes inconvenient in the long run to sustain, because it needs to react to changes in the development and business domain. But developing the test process requires a modest effort, and it also exposes the organization to a possibility of a failed attempt of process development. This effort is not always considered as a productive work, and it generates costs no matter the outcome. The need to develop has to overcome both the acceptable losses from inconveniences in existing process and the justification for the expenses caused by the development effort. This concept is illustrated in Figure 2.

This is what could be expected based on the concern presented by Dybå [5] regarding the status quo mindset. In process development, this could be generalized so that organizations lean towards minimal changes approach, as too radical departures from existing process model are seen not worth the effort. In practice the organizations require a way to compare existing process against possible solutions to understand the next feasible process improvement step. Even completely new concepts have a chance to be adopted, if they resemble or are comparable to the existing process.

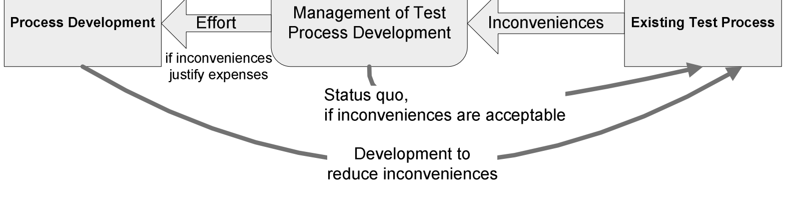

Process Development
Effort
if inconveniences justify expenses
Management of Test Process Development
Inconveniences
Existing Test Process
Status quo, if inconveniences are acceptable
Development to reduce inconveniences

Figure 2. Adopting new practices in test organization.

V. Discussion

The focus of this study was in observing how different test organizations do test process development, and in assessing how feasible the ISO/IEC 29119 [4] standard process model would be in practice. The results indicate that the organizations do mainly sporadic process development, even if the organization continuously collects project feedback and that the new methods are rarely tried out. Also, the proposed standard model itself is feasible, but the practical application suffers from a number of limitations. The main problem was that the standard model has an extensive number of details, but it offers only vague guidelines for actual implementation. Secondly, organizations considered the standard-defined model rather "top-heavy". Particularly the continuous development of the process differed from the industry practices. In many organizations the test completion reports were done, but process changes were only done in "if needed"-basis. Only one of the organizations was definite on trying out new ideas, while all the other organizations had varying doubts. This corresponds to the literature review results, making it evident that organizations aim to preserve the status quo.

In a grounded theory study, the objective is to understand the phenomena which are under observation, and identify a core category which can be explained with all the related categories. Based on these findings, the core category may be extended to a series of observations called hypotheses, and developed to a model – a grounded theory – that can explain the phenomena. In grounded theory studies, the grounded theory generalizes on the basis on what is established in the study. Outside the study, it should be regarded more likely as a general guideline [14].

As for the other limitations and threats to the validity, Onwuegbuzie and Leech [15] have made an extensive framework of different types of threats to validity in qualitative studies. In our work, the issues were addressed by applying several methods; the questionnaire was designed by three researchers to avoid personal bias, feedback on the questions was collected from colleagues to maintain neutrality and from a test interview, the data was collected by researchers so that interviewees understood the questions and finally, in the data analysis, additional researchers who did not participate on the design of the interviews were used to get fresh perspectives on the studied concepts.

VI. Conlusions

In this paper we have presented the results of our study regarding the test process development process and adoption of new testing methods. The results indicate that the organizations do test process improvement mainly sporadically, even in the organizations where the management receives feedback from completed projects. In several organizations the adoption process for new testing techniques is in practice limited to small changes and improvements, as organizations tend to maintain the status quo, unless the process is clearly in need of larger changes.

Besides process development, we also conducted a feasibility test on the ISO/IEC 29119 standard model [4]. Based on the results, it seems that the model itself is feasible, although it contains some concern which should be addressed. Many organizations thought that the fundamentals of the model are sound, but the overall model is "top-heavy" and unnecessarily detailed.

An implication to future research from this study is that organizations need guidelines or a reference model of the standard. By designing such a framework, organizations developing their test processes could have a more realistic view on their existing test process, and have support on deciding objectives in their next test process improvement.

Acknowledgments

This study was supported by the ESPA-project (*http://www.soberit.hut.fi/espa*), funded by the Finnish Funding Agency for Technology and Innovation and by the companies mentioned in the project web site.

References

[1] Kit E. (1995). Software Testing in the Real World: Improving the Process. Reading, MA: Addison-Wesley.

[2] P. Abrahamsson, "Commitment development in software process improvement: critical misconceptions", Proceedings of the 23rd International Conference on Software Engineering, Toronto, Canada, pages 71-80, 2001.

[3] T. Dybå, "Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context", Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering, pages 148-157, Helsinki, Finland, 2003. doi: 10.1145/940071.940092

[4] ISO/IEC JTC1/SC7, ISO/IEC WD-29119, Software and Systems Engineering —— Software Testing, 2010.

[5] J. Kasurinen, O. Taipale and K. Smolander, "Test Case Selection and Prioritization: Risk-Based or Design-Based?", Proceedings of the 4th Symposium on Empirical Software Engineering and Measurement (ESEM), 16.-17.9.2010, Bolzano, Italy, 2010.

[6] V. Kettunen, J. Kasurinen, O. Taipale and K. Smolander, A Study on Agility and Testing Processes in Software Organizations, International Symposium on Software Testing and Analysis (ISSTA 2010), 12.7.-16.7.2010, Trento, Italy, 2010. DOI: 10.1145/1831708.1831737

[7] TMMi Foundation, "Test Maturity Model Intergration (TMMi)", Version 2.0, 2010.

[8] B.C. Hardgrave and D.J. Armstrong, "Software process improvement: it's a journey, not a destination", Communications of the ACM, Vol. 48(11), pages 93-96, 2005. doi: 10.1145/1096000.1096028

[9] EU , "SME Definition", European Comission, 2003.

[10] ISO/IEC, ISO/IEC 15504, Information Technology - Process Assessment, 2002.

[11] B. Glaser and A.L. Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research. Chicago: Aldine, 1967.

[12] A. Strauss A. and J. Corbin, Basics of Qualitative Research: Grounded Theory Procedures and Techniques. SAGE Publications, Newbury Park, CA, USA, 1990.

[13] B.G. Glaser, "Constuctivist Grounded Theory?", Forum: Qualitative Social Research (FQS), Vol 3(3), 2002.

[14] C.B. Seaman, "Qualitative methods in empirical studies of software engineering", IEEE Transactions on Software Engineering, vol. 25, pp. 557-572, 1999.

[15] A.J. Onwuegbuzie and N.L. Leech, "Validity and Qualitative Research: An Oxymoron?", Quality and Quantity, Vol 41(2), pages 233-249, 2007. DOI: 10.1007/s11135-006-9000-3.

Publication VI

Exploring Perceived Quality in Software Organizations

Kasurinen, J., Taipale, O., Vanhanen, J. and Smolander, K. (2011), Proceedings of the Fifth IEEE International Conference on Research Challenges in Information Science (RCIS), May 19-21 2011, Guadeloupe - French West Indies, France, doi: 10.1109/RCIS.2011.6006823

Exploring Perceived Quality in Software Organizations

Jussi Kasurinen[1], Ossi Taipale[1], Jari Vanhanen[2] and Kari Smolander[1]

[1]Software Engineering Laboratory
Lappeenranta University of Technology
Lappeenranta, Finland
jussi.kasurinen | ossi.taipale | kari.smolander@lut.fi

[2]Software Business and Engineering Institute
Aalto University
Espoo, Finland
jari.vanhanen@hut.fi

***Abstract*— Software projects have four main objectives; produce required functionalities, with acceptable quality, in budget and in schedule. Usually these objectives are implemented by setting requirements for the software projects, and working towards achieving these requirements as well as possible. So how is the intended quality handled in this process of pursuing project goals? The objective of this study is to explore how organizations understand software quality and identify factors which seem to affect the quality outcome of the development process. The study applies two research approaches; a survey with 31 organizations and in-depth interviews with 36 software professional from 12 organizations for identifying concepts that affect quality. The study confirms that the quality in software organization is a complex, interconnected entity, and the definitions of desired and perceived quality fluctuate between different process stakeholders. Overall, in many cases the software organizations have identified the desired quality, but are not communicating it properly.**

Keywords- software quality, quality characteristics, quality goals, mixed method study

I. Introduction

Software quality is a composition of different attributes, with the importance of these attributes varying between different types of software products. For example, the desired or important quality characteristics between a game on a mobile phone and control software of an airplane surely have a big difference. How do organizations actually perceive what the quality they require from their products is and what aspects in the development and testing affect the perceived quality outcome?

The main objectives of software engineering include reduction of costs and improvement of the quality of products [1]. To reach the quality objectives in the product, an organization needs to identify their own quality i.e. those quality characteristics which are important for them. After identifying their preferred quality, the next action would be to find the factors in development and testing, which affect these quality characteristics, and ensure they work as intended.

A model that in this sense attempts to specify the different characteristics of quality is the revised software product quality model, as introduced in the forthcoming ISO/IEC 25010 standard [2]. According to the standard, software quality expresses the degree to which the software product satisfies the stated and implied needs when used under specified conditions. In the model, quality consists of eight characteristics, which are functional suitability, reliability, performance efficiency, operability, security, compatibility, maintainability, and transferability. These characteristics are further divided into 38 subcharacteristics, such as accuracy or fault tolerance, which aim to define the quality in measurable terms. In addition, in software business the quality is related both to the development and testing. In the ISO/IEC 29119 standard [3], software test process is defined to comprise of layers, such as organizational test level and test management level. In our study, these standards describe the research subject, software product quality and software testing in organizations.

Testing has a big influence on quality in software business. Testing is also one of the biggest expenses in software development [4]. In one estimate [5], software producers in United States lose annually 21.2 billion dollars because of inadequate end-product quality. Because of the economical importance of software quality, it is important to understand how organizations understand software quality and how organizations decide on quality requirements. The identification of how organizations perceive quality, i.e. which quality characteristics they consider important, and how the quality requirements are catered, helps them to concentrate on essential parts when improving process outcomes from the viewpoint of quality assurance.

However, this task is not easy, as the development and test processes include many concepts which all have possibility to affect the quality in practice. There are several viewpoints by different process stakeholders, with a different perception on what are the important quality characteristics. In this study we explore these concepts and viewpoints in different types of software organizations to understand *how software development and testing affect the perceived quality of end-product* and *which process activities have a major impact on the perceived software quality outcome*.

The paper is structured as follows. First, we introduce comparable studies and related research in Section 2. Secondly, the research process with the quantitative survey method and the qualitative grounded theory method are described in Section 3. The results of the study are presented in Section 4. Finally, discussion and conclusions are given in Sections 5 and 6.

II. Related Research

Software quality is defined in the software product quality standard ISO/IEC 25010 as a combination of different quality characteristics such as security, operability and reliability. However, it is evident that there are also several different approaches on studying quality and quality concepts in the software engineering. So how can something that has so abstract definition as quality be measured or defined for research?

For example, Garvin [6] has discussed the definitions of quality and made extensive definition work for establishing what the quality actually is and how it affects product concepts such as profitability or market situation. Garvin defines five different definitions for quality; transcendent, product-based, user-based, manufacturing-based and value-based definition. Even though they define the same phenomena, product quality, they vary greatly. For example, transcendent quality is "innate excellence", which is an absolute and uncompromising standard for high achievement that cannot be precisely defined, but surely is identified if present. On the other hand, user-based quality is the more common "satisfies user needs" definition, whereas the manufacturing-based definition promotes conformance to the product requirements. Garvin also discusses the different definitions by mentioning that it also explains why different people seem to have different opinion on what is quality; they tend to apply the definition they are most familiar with.

The different aspects and definitions of quality also mean that the measurement of software quality has some considerations. A paper by Jørgensen [7] introduces three assumptions for establishing measurement for software quality: there are no universal quality measurements but meaningful measures for particular environments, secondly, widely accepted quality measurements require maturity in research, and thirdly, quality indicators predict, or indirectly measure quality. In short, Jørgensen establishes that there are no universal measurements, but the approaches using quality indicators – characteristics and attributes – can be used to approximate or predict software quality. Given the perspective of our study, this is in line with our approach of observing and studying the perceived quality and quality-affecting aspects of software process.

Based on the Jørgensen [7] discussion concerning quality indicators and discussion regarding definition of quality by Garvin [6], it seems that applying the classification used in ISO/IEC 25010 would be feasible measurement method. For the survey and qualitative study we also decided to apply literature review to identify different software process activities which would be interesting from the viewpoint of quality. These activities would be called seed categories [8] for the study and form the basis for the survey questionnaire.

For the compilation seed categories [8] in testing, we applied our earlier research results and observations [9] in test processes. Based on our prior research and for example, study by Hansen et al. [10], it is evident that the business orientation affects the testing organization: product oriented organizations should adopt a formal planned testing process and service oriented organizations should adopt a flexible testing process. If the business orientation has an influence on a testing organization, does it have a similar influence on perceived end-product quality? To study this, the construct product/service orientation, was accepted to the seed categories. In addition, Lin et al. [11] also state that quality problems are not only a function of the product or service itself, but also of the development processes. Therefore, constructs describing the development and testing processes and overall process environment were included in this study.

A paper by Boehm and Turner [12] discusses how the applicability of agile [13] or plan-driven methods depends on the nature of the project and the development environment. Boehm and Turner have developed a polar chart that distinguishes between agile methods and plan-driven methods. Abrahamsson et al. [14] writes that agile thinking emerged because software intensive systems were delivered late, over budget, and they did not meet the quality requirements. Therefore the influence of the software development method on perceived quality characteristics was included to the topics of interest.

According to Kit [4], the size and the criticality of the systems among other things emphasize software testing. Also Boehm and Turner [12] select criticality as one of factors affecting the choice of the software development method. Therefore criticality was accepted to our seed categories to see whether it has also an effect on the perceived end-product quality, or preferred quality characteristics.

Guimaraes et al. [15] discusses customer participation in software projects. Customer participation seems to improve specifications of the system and thereby it assists project towards satisfactory outcome. Customer participation and trust between customer and supplier were accepted to the categories to explore their influence on perceived quality.

Based on the literature and our previous studies [9,16,17] we understand that there is a multitude of feasible approaches on studying quality and the concepts that could explain the quality in software processes. Therefore identifying the process activities, which have strong impact on quality outcome, would be complicated. Different organizations, even projects within one organization, may weigh quality characteristics differently and the product quality seems to be related to several, if not all, software engineering concepts in some level.

III. RESEARCH METHOD

Based on literature research, the assessment of quality factors and collecting comparable data on perceived quality in varying organizations was known to be difficult. We decided to approach the problem by applying methods to obtain both statistical and observational data from the organizations, from several viewpoints of software development.

Table 1: Description of the OUs participating in the study

OU	Business	Company size[b] / Operation	Participation
Case A	Modeling software developer	Large / International	Survey, Interviews
Case B	MES[a] producer and logistics service systems provider	Medium / International	Survey, Interviews
Case C	ICT consultant	Small / National	Survey, Interviews
Case D	Maritime software system developer	Medium / International	Survey, Interviews
Case E	Internet service developer and consultant	Small / National	Survey, Interviews
Case F	Safety and logistics system developer	Medium / National	Survey, Interviews
Case G	Financial software developer	Large / National	Survey, Interviews
Case H	ICT developer and consultant	Large / International	Survey, Interviews
Case I	Financial software developer	Large / International	Survey, Interviews
Case J	SME[b] business and agriculture ICT service provider	Small / National	Survey, Interviews
Case K	Logistics software developer	Large / National	Survey, Interviews
Case L	MES[a] producer and electronics manufacturer	Small / National	Survey, Interviews
Other 19 Case OUs	Varies: from software service consultants to organizations developing software components for their own hardware products.	Varies	Survey

[a]Manufacturing Execution System [b]as defined in [20]

We decided to apply two different approaches to validate our own data and further enable us to confirm our findings. To achieve this, we designed a theme-based interview and a survey to collect data on quality concepts; the survey collected data on several organizations to gain a perspective on the industry field as a whole, while the interviews collected considerations of the individual organizations. In the survey, we collected data from 31 software organizations, and in the interviews, we interviewed 36 software professionals from 12 different organizations, in topics such as the test process, test methods and quality in testing. The contacted organizations are summarized in Table 1 and the data collection rounds in Table 2. The themes of the interviews and the questionnaire forms are available at http://www2.it.lut.fi /project/MASTO/.

Combining quantitative and qualitative analyses is a form of methodological pluralism. Methodological pluralism means that the applied study approach does not apply one "correct" method of science, but many possible methods that complement each other [18]. The results of the phases were compared to each other to enable additional validation of the soundness of the data and the analysis. In addition, the population of the study was observed in organizational unit (OU) level. The standard ISO/IEC 15504 [19] specifies an organizational unit as a part of an organization that is the subject of an assessment. An organizational unit deploys one or more processes that have a coherent process context and operates within a coherent set of business goals. An organizational unit is typically a part of a larger organization or company, although in small businesses, the organizational unit may include the entire company. This way the comparison between large, multinational company and small, local, operator became feasible for the purposes of this study.

A. Data Collection

For the interviews we had selected 12 OUs, which represented different software domains, company sizes [20] and operating scales. These 12 organizations were collected from our industrial partners, and supplemented with additional organizations by researchers to represent different types of software business. The selection criteria were that the OU produced software products or services, or offered software-production related services as its main source of income, in a professional and commercial manner. We also accepted only one OU per each company to avoid bias of over-weighting large companies or causing bias from certain types of business practices

All the interviewed case organizations also participated in the survey, for which 19 additional organizations were

Table 2: Organization of data collection rounds

Collection phase	1) Semi-structured interview	2) Structured survey with Semi-structured interview	3) Semi-structured interview
Number of participants	12 focus OU interviews	31 OUs, including 12 focus OUs	12 focus OU interviews
Participamt roles	Designer or Programmer	Project- or Testing manager	Tester or Programmer
Description of participants	The interviewee was responsible for or had influence in software design.	The interviewee was responsible for sofware project or testing phase for software product.	The interviewee was dedicated tester or was responsible for testing the software product.
Focus themes	Design- and Production methods, Testing strategy and -methods, Agile methods, Standards, Outsourcing, Perveiced quality	Test processes and tools, Customer participation, Quality and Customer, Software Quality, Testing methods and -resources	Testing methods, Testing strategy and –resources, Agile methods, Standards, Outsourcing, Test automation and –services, Test tools, Perceived quality, Customer in testing

selected to enhance the statistical relevance. The selection of supplemental OUs was based on probability sampling, randomly picking organizations out of our contacts. The final selection was confirmed with a phone call to check that the OU really belonged to the specified population. Out of the contacted 30 additional OUs, 11 were rejected because they did not fit the population criteria despite of the source information.

The data collection sessions for the survey and interviews lasted approximately an hour and they were recorded for further analysis. The interviewees were selected based on the recommendations from the OU, an emphasis being on the responsibilities and job description of the employee. Additionally, we required that the interviewees should be working in the same project team, or contribute to the same software product, in addition of working in the same OU. In two out of the 36 qualitative interviews, the interviewed organization opted to select two persons for interview, as they considered that they did not have a sufficiently experienced or otherwise suitable individual worker at their disposal. The interviewees were also allowed access to the interview questions before the actual interview. We also did not forbid discussion between prior interviewees nor did we encourage it. Additionally, in one occasion in the first phase we allowed the OU to supplement their first round answers as the interviewee had thought that the given answers lacked relevant details. The data collection was done by three researchers during winter of 2008 to summer 2009.

Structurally, the interviews were implemented with a list of semi-structured questions regarding software testing, quality concepts and software process-themed questions. The interviews included such themes as development methods, agile practices, test resources, test automation and perceived quality. The themes were also related to the set of seed categories [8], which contained essential stakeholders and leads from the literature review [21]. Our aim was to further develop these seed categories based on the observations made in organizations to include the practical aspects that effect software quality.

The first round of interviews included software designers. Our intention was to test whether our prior studies and the observation made on software processes (for example [16, 17]) were still valid. Another objective was to see if our seed categories for this study were selected so that they would yield relevant results.

In the second round interviews the project and test managers were targeted with both qualitative and quantitative instruments. The twelve OUs participating in the first and third round of the qualitative analysis also participated on the survey, which was supplemented with qualitative themes. During the second round, our objective was to collect data on the organization as a whole, as our interpretation was that the managers were in a better position to estimate organizational concepts such as policy effects, overall process, and quality concerns, in contrast to the desired situation.

The third interview round focused on software testers. During this interview round, the focus was on the software testing phases, testing tools and quality aspects in the testing work, further discussing some of the second round topics. Based on the answers we were able to analyze the practical testing work and the effect of quality aspects to the actual testing work.

B. Data analysis on survey

In the quantitative part of the study, the survey method described by Fink and Kosecoff [22] was used as the research method. For the selected approach, methods of data analysis were partially derived from Iivari [23], while the design of the survey instrument was done by the principles derived from Dybå [24]. We used Cronbach alpha [25] for measuring the reliabilities of the constructs consisting of multiple items, and studied the correlations between software quality and other relevant constructs by using Kendall's *tau_b* correlation [26].

Related surveys can be categorized into two types: Kitchenham et al. [27] divide comparable survey studies into exploratory studies, from which only weak conclusions can be drawn, and confirmatory studies, from which strong conclusions can be drawn. This survey belongs to the category of exploratory, observational, and cross-sectional studies.

C. Data analysis on interviews

In the qualitative study we decided to apply the grounded theory method [28, 29, 30]. The grounded theory was first conceived by Barney Glaser and Anselm Strauss [29], but later the original method has diversified into two distinct approaches, introduced in later publications by Glaser [31], and by Strauss and Corbin [30]. The Glaserian grounded theory focuses on observing activities within the environment and relies on emergence that cannot be made fully systematic, whereas Strauss-Corbin is more geared towards systematic examination and classification of aspects observed from the environment. The number of participating organizations, the limited ability to non-intrusively observe the developers while working, and the large amount of data generated by the organizations meant that for classifying and analyzing the data, the Strauss-Corbin approach was considered more feasible to implement in this study.

The grounded theory method has three phases for data analysis [30]. These methods are open coding, where the interview observations are codified and categorized. In this phase, the seed categories are extended with new categories which emerge from the data. It is also possible to merge or completely remove the categories that are irrelevant to the observed phenomena. During the open coding, 166 codes in 12 categories were derived from the 36 interview recordings.

The second phase is called axial coding, in which the relations between different categories and codes within categories are explored. In this phase, the focus is on the inter-category relationships, although some necessary adjustments like divisions or merges may be done to the categories.

The last and third phase of grounded analysis is the selective coding. In selective coding, the objective is to define the core category [28, 30], which explains the observed phenomena and relates to all of the other defined categories. However, in some cases the core category can also be a composition of categories in case one category does not sufficiently explain all the observed effects. In addition, the results may yield useful observations, which explain the observed phenomena, even extending to a model to define the observed activities. In this study, the core category can be characterized as such "umbrella category", which we named as *The Effect of Different Software Concepts on Quality*. We further described the category with five observations that explore the different software process activities and their effect on end-product quality in development process. Finally, based on the study results, we summarize the findings as a grounded theory on feasible approach on enhancing end-product quality.

IV. RESULTS

In this chapter, we present the results from both parts of the study. First we begin with the survey results and then discuss the grounded theory analysis.

A. Results of the quantitative analysis

The questionnaire was divided based on the major themes of the overall study; general information of the organizational unit, processes and tools, customer participation, and software quality. We were able to calculate several different constructs, which were then tested for feasibility and reliability with Cronbach alpha (results in Table 3) and Kendall's *tau_b* (results later in Table 4) tests. Complete survey instrument is also available at http://www2.it.lut.fi /project/MASTO/. In the following, we present the constructs, which were confirmed to affect the perceived quality outcome.

1) Building quality in software process

The interviewees were asked to give their in-sight of two claims, *quality is built in development* and *quality is build in testing*, to estimate which is the source of the quality in their products. This item also included an assessment of the ISO/IEC 29119 test levels in existing organizational processes. The standard was estimated by the maturity levels – the appropriateness of the process compared to the process needs – of different test process levels comparable with the definitions of the standard. These levels, *organizational test policy*, *organizational test strategy*, *project test management level*, and *test execution*, measured the sophistication of the current test process in the OU. Based on maturity estimates, the construct *Existing* p*rocess conformance with the testing standard model* was calculated to describe the existing level of the structures similar or comparable with the ISO/IEC 29119 standard process [3] levels. The used scale was a 5-point scale [21] where 1 denoted "fully disagree" (this level is very bad in our organization) and 5 denoted "fully agree" "this level is very good in our organization). The results based on answers are presented in Figure 1.

According to the results, interviewees emphasized that the quality is built in development (4.3) rather than in testing (2.9). Also for the standard, the results are mostly

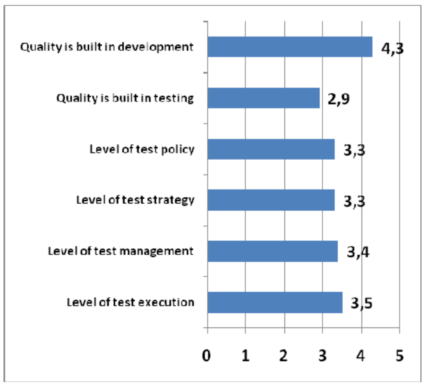


Figure 1. Origin of quality and the realization of the software testing standard ISO/IEC 29119

ambiguous in all test process layers, but slightly favor the lower level activities like management and test execution level.

2) Customer participation

The second survey topic was connected to customer participation. This construct, *Customer participation*, described how customers participated in development and testing processes. For customer participation, the constructs were calculated by summing up the answers of the items and dividing the sum by the number of items. From this group, *Customer participation in the general control*, i.e. in the process steering and in decision making in development, reached acceptable Cronbach alpha value only with two items. These items were *our most important customer reviews project management schedules and progress reports made available by us*, and *our most important customer provides domain training to us*. The Cronbach alpha values for these constructs, amongst the other constructs, are listed in Table 3.

Table 3. The reliabilities of different constructs (acceptance level of >0.7)

Variable	Cronbach alpha
Existing process conformance with the testing standard model	.894
Customer participation during the specification phase of the development.	.855
Customer participation during the design phase of the development.	.772
Customer participation during the testing phase of the development.	.742
Customer participation in the general control.	.702
Trust between customer and supplier.	.699
Elaboration of the quality attributes.	.818

Additionally, the construct *Trust between customer and supplier* described the confidence that the behaviour of another organization will conform to one's expectations as a benevolent action. For measuring this construct, the questions were derived from Benton and Maloni [32]. When calculating the Cronbach alpha for the construct *Trust*, an

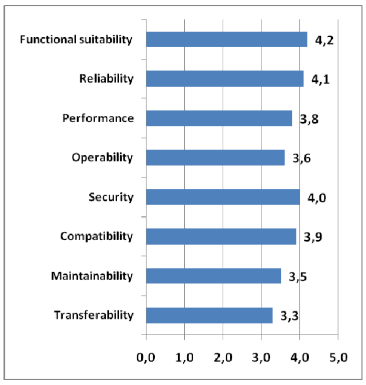


Figure 2. Assessment of fulfilling the different quality characteristics in the end-product

acceptable level was reached with items *our most important customer is concerned about our welfare and best interests* and *our most important customer considers how their decisions and actions affect us*.

3) Quality characteristics and perceived quality

For the third interview topic, interviewees were asked to evaluate the competence level of each ISO/IEC 25010-quality characteristic in their software by a 5-point scale where 1 denoted "this characteristic in our software is taken into account very badly" and 5 denoted "this characteristic in our software is taken into account very well". Interviewees were also advised to leave the attribute unanswered ("this characteristic is irrelevant to our product") if the attribute was not valid for the OU, i.e. if the attribute was irrelevant for the organization. If an organization gave some attribute a high score, it meant that the organization thought that this particular quality characteristic was handled well in the product design and development. The resulting average indicated the perceived level of quality of the organization's product: if organization gave high points to quality characteristics, it was understood that the organization considered their end-product of high quality, if low scores, the organization considered that their product was low quality, or at least not as good as it should be. These results were also used as a construct *perceived overall quality by the organization*. The mean values for all surveyed quality characteristics are included in Figure 2.

Quality characteristics *functional suitability*, *reliability*, *security*, and *compatibility* reached the highest scores, meaning that they were the most well-attended quality characteristics. Even if the results did not vary much (between 3.3 and 4.2), it was indicative that some of the characteristics were generally less attended than others. However, overall all of the attributes were considered at least somewhat important; only in 9 cases (3.6% out of 248 characteristic assessments) the organization considered the assessed characteristic "irrelevant" to their product.

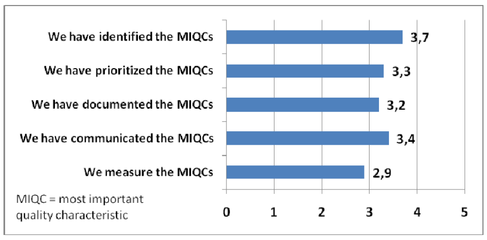


Figure 3. Elaboration of the quality characteristics

In addition of perceiving quality characteristics, the interviewees were asked to evaluate how their organizations elaborated and communicated their quality characteristics. The interviewees were asked to give their in-sight of five claims; *we have (1) identified, (2) prioritized, (3) documented, (4) communicated, (5) we measure the most important quality characteristics.* The construct, *Elaboration of the quality characteristics*, was calculated as the mean of the answers to the claims. Almost all organizations had at least identified their most important quality characteristics (3.7), while measurement and collection of the metrics was not as common (2,9). The results, how organizations elaborate their quality characteristics, are given in Figure 3.

4) Factors for achieving quality characteristics

The effect of different survey constructs was further explored to see how they would correlate with the perceived overall quality of end-product. To achieve this, the Kendall's *tau_b* correlations were calculated between constructs, which were first tested with Cronbach alpha. As based on the Kendall's *tau_b*-analysis, the constructs *Existing process conformance with the testing standard model*, *Elaboration of the quality attributes*, and *Trust between customer and supplier* seemed to positively correlate with the construct *perceived overall quality by the organization* at the 0.01 level. In addition, the influence of some constructs, such as *Customer participation during the design phase of the development*, and *Customer participation in the general control* were almost significant.

Several other constructs were calculated from the data, such as *Software development method* and *Criticality of the OU's end products*, but they did not reach significant correlations and therefore were discarded. The correlations of constructs which had a significant correlation are presented in Table 4, which also includes some insignificant constructs as an example.

Based on these results, the most important factors for achieving better end-product quality and pursuing quality characteristics in software products are closely related to the development process maturity and elaboration of quality goals. Those organizations that had identified and communicated their most important quality characteristics or were confident with their development and test processes were also confident with the levels of their quality characteristics. The organization which thought that the appropriateness and sophistication of their development process was high or said that the identification of the important quality attributes was in a high level, also considered that their products implemented the quality characteristics well. Also aspects such as customer participation in certain parts of the development and trust between stakeholders were also observed to be beneficial.

Table 4. Correlations between different surveyed constructs and the perceived overall quality

Construct	Kendall's tau_b
Software development method	-.195 .158
Criticality of the OU's end products	.171 .226
Existing process conformance with the testing standard model	.505 ** .000
Customer participation during the specification phase of the development.	.120 .377
Customer participation during the design phase of the development.	.231 .092
Customer participation during the testing phase of the development.	.141 .287
Customer participation in the general control.	.261 .057
Trust.	.436 ** .002
Elaboration of the quality characteristics.	.437 ** .001

Kendall's correlation (N=31)
** Correlation is significant at the 0.01 level (2-tailed).

B. *Results from the grounded analysis*

The grounded theory analysis data was collected from 36 interviews held at 12 software producing organizations. We interviewed software designers, project or test managers and testers in three phases. This data was then codified and analyzed, which led to the definition of several categories and factors that were observed to have an effect on the perceived quality and quality output of the product development process, or based on the literature were considered important. In the following sections, we introduce these categories and observations.

The core category, "*The Effect of Different Software Concepts on Quality*", is defined as a composition of seven other categories. These categories were collected from the topics that interviewees mentioned regularly when discussing about the quality aspects and perceived quality in their software process. For example, standardization and the role of the customer in the process were mentioned regularly. In some occasions a category was included to test the possible lead-ins from the survey and literature reviews. For example, the effect, or more precisely the lack of effect, for concepts such as the product/service-orientation or criticality was studied more closely in the qualitative study. A summary of the categories is shown in Table 5.

The category "*Desired Quality Characteristics in design"* explains the ISO/IEC 25010 quality definitions that were considered to be the most important characteristics from the viewpoint of software designers. These quality aspects were most likely those that were used in software process, especially in specification, design and other early development phases. For comparison, the second category "*Desired Quality Characteristics in testing*" explained the quality characteristics that were considered to be the most important from the viewpoint of testers, and subsequently also those, the testing work focused on.

The category of "*Level and Effect of Criticality"* is a two-fold category. First is the level indicator of criticality of the product the interviewed OU is developing. The criticality level scales from 1-5. In this scale, 5 is the highest level meaning "may cause loss of human life", 4 is "may cause bodily harm or great economical losses", 3 "significant economical losses", 2 "small economical losses" and 1 "no effect or user irritation". The scale is similar to other criticality measurements, discussed for example in [32]. The latter part of the category is assessment on how the test process would change if the criticality level of their product increased.

The category of "*Effect of Customer*" defines the effect the customer has on the end-product quality. This category defines the most influential action or generally the possibilities of the customer to affect the quality, in actions such as "extend deadline" or "allow larger budget". It should be noted that in this category, the most potent effect may also be harmful for the end-product quality, like limiting access to the final working environment or requesting unfeasible changes.

The category of "*Applied standards"* lists the standards the organizations are using in either development or test

Table 5: Categories from qualitative analysis

Category	Description
Desired Quality Characteristics in design	The quality characteristics that the software designers consider important.
Desired Quality Characteristics in testing	The quality characteristics that the software testers consider important.
Level and Effect of Criticality	The level of end-product criticality and the effect of how increased criticality would affect testing work.
Effect of Customer	The effect of customer to the end-product quality, and short description on what constitutes this effect to take place.
Applied Standards	The standards that are officially followed and enforced in either software development or test process of the organization.
Effect of Outsourcing	The effect of outsourcing development process activities has to end-product quality, and short description of what constitutes this.
Product/Service-orientation	The distribution of business in interviewed organization divided between product and service-oriented activities. Assessed by the managers.

process. Even though many organizations applied parts of the standards, or followed the process efficiency unofficially by measures derived from standards, in this category only the fully applied, systematically used, standards and competence certificates are listed.

The category of "*Effect of Outsourcing"* defines the effect outsourcing has on the perceived quality of an end-product, including effects like allowing more focus on core products or critical aspects. This category defines the most influential way the outsourcing affects the process outcome quality.

The category of "*Product/Service-orientation"* represents the ratio between product-oriented activities and service-oriented activities in an OU. This approximation is directly taken from the survey interviews with managers.

1) Observations from the data

The observations were developed based on the categories. These observations either define software development concepts that affected the perceived quality in a software process, affected quality in a product or were considered an important item for composing more complex constructs of software quality. All these observations are based on the findings made during the analysis. The summary of the findings which were the basis for the observations, are available in Table 6.

Observation 1: The importance of quality attributes vary between different process stakeholders in an organization.

The first finding confirmed, as suggested by the literature [6], that the conceptions of quality vary even within one project organization. The software designers and testers were asked to rank the ISO/IEC 25010 quality attributes in the order of importance. Although the testers and designers were working on a same organization, the most important attribute was the same only in four case organizations (A, D, K and L) out of twelve. All the same attributes, although not necessarily in the same order, were mentioned by two organizations, cases L and D.

It seems that the designers were slightly focused towards usability aspects such as operability or functional suitability, while testers were focused towards technical attributes like security or reliability, meaning that each participant had a different view on what quality should be.

Observation 2: The effect of product/service-orientation or criticality on the importance of quality attributes is low.

The product-oriented organizations and service-oriented organizations seem to have similar priorities in quality attributes. For example, Case E, which is a fully service-oriented software producer, promotes the same attributes as Case F, which is mostly product-oriented. Similarly Case G, which has a large emphasis on the service-orientation, has

Table 6: Observations from the case organizations

	DQC[a] in design	**DQC[a] in testing**	**Level and Effect of Criticality**	**Effect of Customer**	**Applied standards**	**Effect of Outsourcing**	**Product/ Service orientation**
Case A	Functional suitability, Reliability	Functional Suitability, Security	2: Security, Reliability get more attention	Enhances the quality by participating closely.	ISO9000-seires, ISTQB-certificates for testers	No meaningful effect.	100% product
Case B	Functional suitability, Performance efficiency	Perfomance efficiency, Operability	3: Performance efficiency, Functional suitability get more attention	Enhances the quality by allowing larger expenses.	CMMi	-	100% product
Case C	Maintainability, Functional suitability, Reliability	Reliability, Operability	4: Central aspects get more attention.	Enhances the quality by providing feedback	ISO9000-series, ISO12207	May weaken the quality by causing instability in the process.	80% product, 20% service
Case D	Functional suitability, Operability	Functional suitability, Reliability, Operability	4: Security gets more attention	Enhances the quality by providing feedback	Officially none	May enhance quality.	55% product, 45% service
Case E	Operability, Security	-	1: Central aspects get more attention	Enhances the quality by allowing larger expenses.	Officially none	May weaken the quality by causing instability in the process.	100% service
Case F	Operability, Functional suitability	Reliability, Compatibility, Security	5: Central aspects get more attention	Enhances the quality by participating closely.	ISO9000-series, domain-based certifications.	May weaken the quality by causing instability in the process.	83% product, 17% service
Case G	Reliability, Performance efficiency, Security	Functional suitability, Operability, Performance efficiency, Reliability	3: Security, Functional suitability get more attention	Enhances the quality by providing feedback	ISO9000-series SPICE, ISTQB-certificates for testers	-	40% product, 60% service
Case H	Security, Reliability	Maintainability, Operability, Performance efficiency, Reliability	2: Functional suitability gets more attention	Enhances the quality by providing feedback	CMMi, ISTQB-certificates for testers	No meaningful effect.	50% product, 50% service
Case I	Functional suitability, Performance efficiency, Reliability	Security, Compatibility, Functional Suitability, Reliability	4: Functional suitability, Reliability get more attention	Enhances the quality by participating closely.	CMMi	Enhances the quality by allowing focus on critical aspects.	63% product, 37% service
Case J	Performance efficiency, reliability, Operability	Functional Suitability, Reliability, Performance Efficiency	2: Reliability, Functional suitability get more attention	Enhances the quality by providing feedback	Officially none	No meaningful effect.	100% product
Case K	Functional suitability, Reliability, Performance efficiency	Functional suitability, Reliability, Security	3: Central aspects get more attention.	Enhances the quality by allowing larger expenses.	Officially none	Enhances the quality by allowing focus on critical aspects.	100% product
Case L	Functional suitability, Operability	Functional suitability	3: Central aspects get more attention.	Weakens the quality by requiring late changes.	Officially none	May weaken the quality by causing instability in the process.	75% product, 25 % service

[a]Desired Quality Characteristics

the same two most important attributes in both design and testing as Case J, which is fully product-oriented. The interviews reflected this consideration to some degree. The type of software may change between projects, but the development and testing process is done in a similar fashion in every project.

"[The project type is irrelevant] as we make things the same way in any case if we want to keep any quality." – Tester, Case F

"Quality is built in design with test cases [in all projects]." –Tester, Case G

The criticality of the software product seems to have only a small effect on the test process activities. When asked to reflect on how the development priorities of a software project change in the case of a higher criticality, the main features were considered to gain more attention in five organizations. In the other seven organizations, certain quality aspects, such as functional suitability, reliability or security, gained more attention. Overall, the criticality was not considered to cause major process changes in any organizations.

"Security… and reliability, they still are number one; in this business they always are." –Designer, Case G

"Yes, in some cases the security would be concerned" – Tester, Case D

A clear indicator of the effect of criticality was observed when comparing the cases E, F and K. Case K was a completely product-oriented organization with an average criticality, Case E was a completely service-oriented organization with a low criticality and F a high-criticality product-oriented OU. The differences between software products in these organizations can be considered quite large, but yet the effect of criticality was considered similar; the process becomes more rigid but the approach stays the same.

"I think, within our business space, it [testing process] would stay the same" – Designer, Case K

"Activities should always aim to progress work towards objectives [regardless of what we are doing]." –Designer, Case E

"[Security] is something that is always taken into account… but maybe we should focus more on that." – Designer, Case F

Observation 3: The standards in software testing do not affect the quality characteristics, as they are not widely used in practice even though organizations in general are positive towards standardization.

Testing standards and certifications in the case organizations were rarely applied. The most commonly applied standards were CMMi and ISO9000 models, which both focus on general process quality measurements. In five organizations no standards were followed officially, although some method of measuring process efficiency existed in all organizations.

"ISO9000… well officially we do not have any certificates for it, but that is the one we based our own on." – Manager, Case G

"CMMi reviews… as far as I know they, however, have been internal." – Manager, Case H

As for testing-related standards, the application was even more sporadic. Only in three cases, G, H and L, some form of official testing certification was applied.

"We have one tester who has done it [ISTQB]… he trains the other testers. That seems to work for now." – Tester, Case A

"We have this ISTQB. All our testers as far as I know have done it." – Tester, Case H

"We have testers participating in the ISTQB training." – Tester, Case G

Even though many organizations did allow, or were generally positive towards participating on certification training, the number of testers who had actually acquired a formal certification varied. The level of currently applied test-related standards and certificates seems to indicate that organizations could have use for a new testing standard. This was indicated by feedback given by the interviewees when discussing the purposes of the upcoming ISO29119 standard and the standards currently applied:

"It would help us to have some way to organize testing in a smart way. A prepared model would be ideal." – Tester, Case L

Observation 4: The general impact of a customer to the perceived end-product quality is positive, but a customer is required to either provide resources or commit to the project.

The customer in a software project was generally considered to have a positive impact on end-product quality.

"The feedback from the client is important to have." – Designer, Case H

"It is easier to do [good quality] if the customer is involved." – Manager, Case F

"The customer brings their own set of quality requirements… it [quality] becomes everyone's objective." – Manager, Case G

However, to actually have an impact on the quality, the customer was required either to provide a substantial financial contribution to the project, to give relevant feedback or to commit otherwise to the project, offering insight and contributions to the project along its progress.

"If they want high quality [they increase the project budget]" – Designer, Case K

"Giving feedback is the key [to quality]." – Manager, Case J

"Participation to the specification is the first, the second is acceptance testing to see if everything is done as agreed and the third is communication, meaning comments and such…" –Manager, Case A

"The customer has to be active especially in testing and specification phases." – Manager, Case I

On the other hand, one organization also noted that in some occasions the customer may hinder the quality, for example by requiring late or unfeasible changes to the product without providing enough support to allow such operations.

"If a customer wants something really stupid and pays for it, then we have to do it." – Designers, Case L

"In one case, the customer did not allow us to use their systems, so we could not do the final tests." – Designers, Case L

Observation 5: Outsourcing may cause quality issues smaller organizations.

It seems that the OUs from small companies are cautious to apply outsourced resources in their projects. In our study, the small-company originating cases L, C and E all were uncertain or concerned regarding the quality of outsourced resources. They considered outsourced resources and third-party-based software modules hazardous, or at least challenging to implement in their own projects:

"There always seem to be some problems with modules brought from outside." – Designer, Case L

"If you start from scratch when outsourcing, it fails unless a lot of energy is used to assure it." –Manager, Case E

As a contrast, OUs from large companies – cases K, H, I and L – considered outsourcing to be a feasible option. Generally their opinions seemed more positive, even to the extent of considering it to enhance quality by allowing focusing on central aspects of the software.

"In outsourcing, we can require that in our product, we allow only this and that amount of errors… by partnering, we can easily assure quality in those aspects." –Manager, Case K

"They go through the same test process so I don't see any problems with them." –Designer, Case H

"It does not affect. Bought code is reviewed and integrated similarly as our own." – Designer, Case L

It would seem that the OUs from larger companies do gain benefits from belonging to a larger organization, at least in applying outsourced resources. The rationale for this observation may be that large companies have more power or influence; small companies may be unable to pay similar amounts as larger companies, to get exactly what they want, so they experience more problems. Another viable explanation could be that large organizations may have more experience of outsourcing or at least have more resources to organize and administrate the outsourcing activities.

With outsourcing, the effects of open source software modules in professional software products were also discussed by case organizations D, F, H and I. Case F considered open source resources to be useful, as they allowed the development to create secondary features out of existing components. In their organization, the application of open source resources to non-critical parts of the product was considered to improve overall quality.

"The best open source modules follow standards more closely than some commercial products" –Tester, Case F

Cases D and H expressed similar considerations; Case D had implemented some secondary features with open source resources, while Case H was yet to apply open source but was willing to try should something applicable be found. Case I applied some open source along with other outsourced modules, but was unsure if especially the "opensourceness" had any effect on quality.

V. Discussion

One common theme seems to be the confidence in testing process. Both the survey results and qualitative analysis established that there are some indicators which affect the perceived software quality, such as appropriateness of the testing process in relation to the product, to communication of most important quality characteristics or to customer participation to the development process. Along with the appropriateness of test process, the overall level of standardization seems to have positive effect on quality. However, especially in testing, the existing test processes rarely seem to apply standards to a large degree. Even if the overall attitudes towards test standardization and test certification programs are positive, the application of standards in several studied organizations was still at a too low level to actually have a visible influence on the process.

As for other studied process concepts, the software development method, the product/service-orientation of the organization nor the criticality affected the perceived quality to a large degree; it seems that the product quality can be sufficient with any development method, and that the main criteria for quality characteristics comes from the product domain, not from the criticality of the product. Surely the highly critical software goes through more rigorous testing than that with low criticality, but the importance of quality characteristics is not related to criticality. For example, in the case organizations in finance domain, the most important quality characteristics were reliability, security and functional suitability, regardless whether the application itself was used to service individual users or a large network. The criticality level varied between levels of 2 (small economical losses) and 4 (great economical losses), but the quality goals and importance of quality characteristics, stayed the same. Similarly, a software development method, whether it applied agile practices or traditional design-based approach, nor the product/service-orientation, affected the importance of quality characteristics.

One interesting observation was that designers and testers rarely had similar considerations on the "most important quality characteristics". This phenomenon surely has an effect on pursuing quality in the software products, as the elaboration of desired quality did correlate with the improvement of quality. Overall, it seems that the desired quality characteristics are usually not identified nor communicated strongly enough throughout the organizations, as the identified quality characteristics were usually based on personal preferences, similarly as discussed by Garvin [6].

In our study we have established that there are factors which affect the perceived end-product quality. The participation of the customer, defining and communicating the quality objectives, and creating a feasible software process, which has addresses the needs of desired quality, were established to have positive correlation with the perceived quality. Summarizing these findings to one grounded theory, it would seem that *creation of appropriate, systematic test and development processes, promoting active participation from customers, and identifying and communicating the desired quality characteristics through the organization offer a good starting point for pursuing better quality in end-products*.

Applying two different approaches allowed this study to observe the quality from different viewpoints, and overall do comparisons between different sources of data. In this sense, the threats to validity for the results of this study are low, but there are some concerns for the study validity.

First of all, in survey the sample size of 31 organizations may seem somewhat limited. However, similarly as in [23], the sample size is small but sufficient if analyzed correctly. In our study, the threat of overfitting the data - over-representing certain sub-groups of participants - was addressed by selecting the organizations to represent different software domains and types of organizations, and triangulating the data with different approaches. Also in terms of the number of organizations, a paper by Sackett [34] discusses the conceptualization of signal-to-noise-ratio in statistical research. Their approach to define confidence as based in practicality of observations: *confidence = (signal / noise) * square root of sample size*. In practice, this indicates that the confidence for the result being non-random weakens if the amount of noise increases while signal decreases. In the Sackett model, the attributes are abstracted, meaning that the noise can be considered to be uncertainty on any source of data. The concept is that the confidence in the survey data increases the validity of the study. Our study addressed this problem by organizing face-to-face interviews with clients and applied researchers as the interviewers to ensure that the interviewees understood the questions and terminology correctly. Therefore in Sackett terms it can be argued that our signal was very good and noise low, so the overall confidence should be good.

As for the validity of the qualitative parts of this study, there are some threats that should be addressed [35]. For example, Golafshani [36] discusses the validity and reliability of qualitative research, and makes some notions on the topic. First of all, the reliability and validity in a qualitative study are not the same, traditionally mathematically proved concepts, as in a quantitative study, but rather a conceptualization of trustworthiness, rigor and quality of the study. To increase the validity in qualitative study, the research must eliminate bias and remain truthful to the observed phenomena. Similarly, a grounded theory is not a theory in mathematical sense, establishing a universal truth or causality, but rather a generalization of observations, offering guidelines and considerations for best practices when taken outside of the study scope [30].

The concept of research validity has been taken even further by Onwuegbuzie and Leech [37], who create model for threats of validity in qualitative studies. They summarize that in qualitative research the threats to internal validity and external credibility are context sensitive; in quantitative studies the objective is to minimize the amount and effect of invalid data, but in qualitative studies, the threats have to be individually assessed based on truth value, applicability, generalizability and such. As these measurements are interpretative, the validity should be addressed by providing enough documentation on the research process, analysis methods and reasoning for the presented results.

As mentioned in the study by Jørgensen [7], in measuring and comparing quality, there are no universal measurements. There is only a possibility to produce relevant results within the context. Obviously our study has the same limitations, but for our research objectives in observing perceived quality in software development, our intention was to observe and identify software engineering aspects which should be used to define general guidelines on how the quality should be addressed or improved. In this objective, we managed to identify the effect of several components, such as the role of the customers, product criticality, process appropriateness or development method.

VI. Conclusions

In this paper we have presented our multi-method study on observing perceived quality in software organizations. Our results indicate that there are several concepts which affect the perceived software quality, such as customer, outsourcing or communication between stakeholders. On the other hand, it also seems that several process concepts such as criticality, product/service-orientation, development method or open source approach do not have any major effect on the perceived end-product quality. It is obvious that high-criticality products do have fewer faults than those on the low end of the scale, but the desired quality characteristics do not change significantly between the criticality levels. Another important finding was that even within one organization the importance of quality attributes seems to have variation between different stakeholders and viewpoints of the software process.

In the majority of the organizations, the testers and designers considered quite differently of what are the “most important” or “most desired” quality attributes of the product. It seems that the desired objectives, and desired quality, must be communicated clearly to reach every stakeholder in the organization as the desired quality and quality requirements are not obvious, “common sense” aspect. Overall, it seems that generally feasible approach in pursuing better end-product quality would be to create systematic test and development processes, promote active participation of customers and identify and communicate the desired quality characteristics through the organization.

As for future work, it is evident that concepts which in this study were observed to have correlation with perceived quality are also closely related to software process improvement. It would be beneficial to study how these observations could be integrated into a process improvement project, and empirically validate the study-established factors, which had observable effect on the perceived end-product quality.

ACKNOWLEDGMENT

This study was supported by the ESPA-project (http://www.soberit.hut.fi/espa), funded by the Finnish Funding Agency for Technology and Innovation, and by the companies mentioned in the project pages.

REFERENCES

[1] Osterweil L. J. (1997). "Software processes are software too, revisited: an invited talk on the most influential paper of ICSE 9," in International Conference on Software Engineering, Proceedings of the 19th international conference on software engineering, Boston, pp. 540-548.

[2] ISO/IEC (2008). ISO/IEC 25010, Softaware Engineering – Software product Quality Requirements and Evaluation (SQuaRE) Quality Model.

[3] ISO/IEC (2009). ISO/IEC 29119, Software Testing Standard.

[4] Kit E. (1995). Software Testing in the Real World: Improving the Process. Reading, MA: Addison-Wesley.

[5] Tassey G. (2002). The Economic Impacts of Inadequate Infrastructure for Software Testing. U.S. National Institute of Standards and Technology report, RTI Project Number 7007.011.

[6] Garvin D.A. (1984). "What Does "Product Quality" Really Mean?", Sloan Management Review, Issue 4, pages 25-43.

[7] Jørgensen M. (1999). "Software quality measurement", Advances in Engineering Software, Vol 30(2), pages 907-912.

[8] Miles M.B. and Huberman A.M. (1994). Qualitative Data Analysis. Thousand Oaks, CA: SAGE Publications.

[9] Taipale O. and Smolander K. (2006). "Improving Software Testing by Observing Practice", Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE), 21-22 September 2006, Rio de Janeiro, Brazil, IEEE, pp. 262-271.

[10] Hansen M. T., Nohria N. and Tierney T. (1999). "What's Your Strategy for Managing Knowledge?," Harvard Business Review, vol. 77, pp. 106-116.

[11] Lin J., Brombacher A.C., Wong Y.S. and Chai K.H. (2004) "Analyzing Quality Information Flows in Cross-company Distributed Product Development Processes," in Engineering Management Conference, Singapore, pp. 973-977.

[12] Boehm B. and Turner, R. (2003), "Using Risk to Balance Agile and Plan-Driven Methods," Computer, vol. June, pp. 57-66.

[13] Fowler M. & Highsmith, J. (2001), The Agile Manifesto.

[14] Abrahamsson P., Salo O., Ronkainen J. & Warsta J. (2002), Agile Software Development Methods: Review and Analysis. VTT Publications 478.

[15] Guimaraes T., McKeen J. D. & Wetherbe J. C. (1994) The Relationship Between User Participation and User Satisfaction: An Investigation of Four Contingency Factors. MIS Quarterly/December 1994, 26.

[16] Karhu K., Repo T., Taipale O. and Smolander K. (2009). "Empirical observations on software testing automation", IEEE Int. Conf. on Software Testing Verification and Validation, Denver, USA.

[17] Kasurinen, J., Taipale, O. and Smolander, K. (2009). "Analysis of Problems in Testing Practices", Proc. 16th Asia-Pacific Conference on Software Engineering, 1.-3.12., Penang, Malaysia.

[18] Hirschheim R. A. (1985), 'Information systems epistemology: an historical perspective', in R. H. E Mumford, G Fitzgerald, T Wood-Harper (ed.), Research Methods in Information Systems, North-Holland, Amsterdam.

[19] ISO/IEC (2002). ISO/IEC 15504-1, Information Technology - Process Assessment - Part 1: Concepts and Vocabulary, 2002.

[20] EU (2003), "SME Definition," European Commission.

[21] Seaman C.B. (1999). "Qualitative methods in empirical studies of software engineering", IEEE Transactions on Software Engineering, vol. 25, pp. 557-572.

[22] Fink A. and Kosecoff, J. (1985). How to conduct surveys A Step-by-Step Guide. Newbury Park, CA: SAGE Publications, Inc..

[23] Iivari J. (1996). "Why Are CASE Tools Not Used," Communications of the ACM, vol. 39, pp. 94-103.

[24] Dybå T. (2000). "An Instrument for Measuring the Key Factors of Success in Software Process Improvement," Empirical Software Engineering, vol. 5, pp. 357-390.

[25] Cronbach L. J. (1951), "Coefficient Alpha and the Internal Structure of Tests," Psychometrika, vol. 16, pp. 279-334.

[26] SPSS (2010). SPSS 17.0. Chicago: SPSS Inc. http://www.spss.com. Referenced 29.3.2010.

[27] Kitchenham B. A., S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg (2002). "Preliminary Guidelines for Empirical Research in Software Engineering," IEEE Transactions on Software Engineering, vol. 28, pp. 721-733.

[28] Paré G. and Elam J.J. (1997). Using Case Study Research to Build Theories of IT Implementation. The IFIP TC8 WG International Conference on Information Systems and Qualitative Research, Philadelphia, USA. Chapman & Hall.

[29] Glaser B. and Strauss A.L. (1967). The Discovery of Grounded Theory: Strategies for Qualitative Research. Chicago: Aldine.

[30] Strauss A. and Corbin J. (1990). Basics of Qualitative Research: Grounded Theory Procedures and Techniques. SAGE Publications, Newbury Park, CA, USA.

[31] Glaser B.G. (2002), "Constuctivist Grounded Theory?", Forum: Qualitative Social Research (FQS), Vol 3(3).

[32] Benton W. C. & Maloni M. (2004) The influence of power driven buyer/seller relationships on supply chain satisfaction. Journal of Operations Management, 22.

[33] Bhansali, P.V. (2005). "Software safety: current status and future direction", ACM SIGSOFT Software Engineering Notes, Vol 30(1).

[34] Sackett D.L. (2001). "Why randomized controlled trials fail but needn't: 2. Failure to employ physiological statistics, or the only formula a clinician-trialist is ever likely to need (or understand!)". CMAJ 165 (9): 1226–37, October.

[35] Robson C. (2002). Real World Research, Second Edition. Blackwell Publishing,

[36] Golafshani N. (2003). "Understanding Reliability and Validity in Qualitative Research", The Qualitative Report, Vol 8(4), pages 596-607.

[37] Onwuegbuzie A.J. and Leech N.L. (2007). "Validity and Qualitative Research: An Oxymoron?", Quality and Quantity, Vol 41(2), pages 233-249. DOI: 10.1007/s11135-006-9000-3.

Publication VII

A Self-Assessment Framework for Finding Improvement Objectives with ISO/IEC 29119 Test Standard

Kasurinen, J., Runeson, P., Riungu, L. and Smolander, K. (2011), Proceedings of the 18th European System & Software Process Improvement and Innovation (EuroSPI) Conference, Roskilde, Denmark, 27.-29.6.2011, doi: 10.1007/978-3-642-22206-1_3.

A Self-Assessment Framework for Finding Improvement Objectives with ISO/IEC 29119 Test Standard

Jussi Kasurinen[1], Per Runeson[2], Leah Riungu[1] and Kari Smolander[1]

[1] Software Engineering Laboratory, Lappeenranta University of Technology, P.O. Box 20, FI-53851 Lappeenranta, Finland
{jussi.kasurinen, leah.riungu, kari.smolander}@lut.fi
[2] Software Engineering Research Group, Lund University, P.O. Box 118, SE-22100 Lund, Sweden
per.runeson@cs.lth.se

Abstract. One of the latest additions in defining the test process is the upcoming ISO/IEC 29119 standard, which aims to define a universally applicable generic test process model. However, currently the standard does not offer any support for the adoption process of the model. In this paper, we present our framework, which aims to combine a maturity level-based approach with the standard process. Our objective was to create an easy-to-use framework for organizations to assess how closely their existing test process follows the standard, and give feedback on improvement objectives. Our results indicate that combining maturity levels with the standard is a viable approach to assess the implementation of the standard in practice.

Keywords: ISO/IEC 29119, self-assessment framework, test process maturity.

1 Introduction

In an ideal world, every time a new software system is produced, the test phase verifies every possible use case and scenario that can be done with the new system. However, the reality is that in the most cases, this is simply not possible, as comprehensive testing for a complex programs would take too long or consume too many resources [1]. In fact, even with realistic test resources, the test process usually ends up being the most expensive item in the software development project [2], taking as much as half of the development budget. Testing also has a large role in determining whether the end-product is commercially successful or not [3].

Because the test process is important for successful software development, it also means that the test process should be appropriate for the software organization and follow good practices. To help organizations to achieve this objective, the International Organization for Standardization (ISO) is developing a new standard focusing solely on software test process, called ISO/IEC 29119 Software Testing [4].

In this study, the current draft of the standard is examined from the viewpoint of its applicability in organizations. Our objective is to extend the standard with maturity levels, and to define a practical framework for organizations to self-assess their existing test process against the standard. The framework also aims to create a simple process assessment tool, which produces practical objectives on improving the test process and adopting practices as defined in the ISO/IEC 29119 standard.

The method to develop such a framework was to combine the processes defined in the standard test process model with an existing testing-related maturity model. In this study, the levels from Test Improvement Model (TIM) [5] were applied. This maturity model was selected based on the conceptual similarities between the key areas in the maturity model and the test processes defined in the ISO/IEC 29119 standard. The resulting framework was also assessed for its feasibility with a pilot study, in which the framework was used to evaluate four real-life case organizations' existing test processes. In three of these organizations, the profile was also assessed for accuracy and usability by the organization itself to gain feedback and enhancement proposals for further development.

The rest of the paper is constructed as follows: In Section 2, the existing test process maturity models and other related research is presented. In Section 3, the developed self-assessment framework, combining ISO/IEC 29119 processes and TIM levels is introduced. In Section 4, the pilot study with the framework and evaluation of the produced development objectives is discussed. Section 5 discusses the framework and pilot study, while Section 6 brings the paper to a conclusion.

2 Related research

The ISO/IEC 29119 test standard [4] is by no means the first model for defining test processes. The Test Maturity Model integrated (TMMi) [6] defines a maturity model for assessing test processes by applying principles of Capability maturity model integrated (CMMi) [7] framework on a software testing context. Unlike the ISO/IEC 29119 standard, TMMi is structured as an assessment model, where TMMi practices are adopted in several maturity phases. The TMMi-model focuses on iterative improvement processes, where the process maturity level increases as the process improves. However, the application of the TMMi model in real-life organizations suffers from some practical limitations [8,9]. For example, the concept of moving all processes to one level before advancing to next is unrealistic to implement [9], and the level requirements are criticized for being counter-intuitive [8].

Another model, which introduces maturity levels to test processes, is the Test Improvement Model (TIM) [5]. TIM is based on CMM, a predecessor of CMMi [7], and it focuses on developing the test process improvement objectives from the existing state of the test process in the assessed organization. TIM also addresses the limitation of requiring all of the processes to reach one level before moving to the next; the test process is divided to key areas, such as *planning and tracking* or *testware*, which are all assessed individually. The key areas are assessed based on the existing practices, in levels which are *baseline*, *cost-effectiveness*, *risk-lowering* and *optimizing*, in the order of maturity. The model promotes a concept of balanced development, but does not enforce a strict parallel improvement of the key areas.

There are also previous attempts to develop a light-weight assessment framework for software testing. One such example is the model, which is especially geared towards small- to medium-sized [10] businesses, the Minimal Test Practice Framework (MTPF), designed by Karlström et al. [11]. This model elaborates on the concepts presented in the TMMi and TIM, simplifying the assessment process and enabling the framework to be applied in the smaller organizations, which may not have a need for extensive, large models. The MTPF model simplifies the test process into five categories and defines a three-step process for each category to mature towards the better testing practices, especially to align with the growth of the company. The model aims to simplify the process improvement process, offering concrete improvement objectives for small test organizations.

3 Self-assessment framework

The draft for ISO/IEC 29119 standard model [4] aims to specify a generic test process model, which promotes the best practices of the discipline. However, the standard presents a process model, giving no input on how to assess the goals of process improvement or how to achieve conformance with the standard in a real-life organization. The objective of this study is to present a concept-level framework, with the intended purpose of enabling the evaluation of the existing test process against the practices defined in the standard.

3.1 The ISO/IEC 29119 test process model

The current draft of the ISO/IEC 29119 [4] standard model is structured as several layers which have different processes. The topmost layer is the *organization level*, in which the organization-wide test policies and test strategy are defined. Below the organizational level is the *project-level* management, which defines the activities and responsibilities of the management in the individual projects. At this level, the organization-defined documents are used in definition of a test plan, and as a basis for deciding the test completion criteria. The project-level management also oversees the lowest level of the model, the *test execution level*, and reports test completion reports and feedback to the organizational management. At the test execution level the actual testing tasks are completed and reported to the management. The test execution level is further divided to two process types, static and dynamic. Static processes are the testing activities, which are carried out throughout the project, whereas dynamic processes are changing during the project advancement. In the model, these processes, which contain all of the activities, roles and responsibilities, are defined as follows:

- *Organizational test process* (OTP) develops and manages organizational test specifications, such as test policy and test strategy. It also is responsible for monitoring and controlling lower layers of the process.
- *Test management processes* (TMP) are the project-level management activities in the test process. TMP defines the test planning, test monitoring and control and test completion, and is also responsible for maintaining the test plans.

- *Test planning process* (TPP) is the process which is responsible for developing the test plan. Depending on the project phase, this may be project test plan, or test plan for a specific phase.
- *Test monitoring and control process* (TMCP) ensures that the testing is performed in line with the test plan and organizational test documents. It also is responsible for identifying updates necessary for the test plan.
- *Test completion process* (TCP) is the process which includes activities, which are done when testing is completed. It also ensures that the useful test assets are made available for later use.
- *Static test processes* (STP) describes how static testing activities, such as test preparation, test result review or test follow-up are done. These activities are the “general” activities, which are done to all test cases in all test phases of the project.
- *Dynamic test processes* (DTP) describe how dynamic test activities such as test implementation, test execution, test environment set-up and test incident reporting are done in the organization. These activities are the “practical” activities, which vary between different types of testing.

Some of the processes (like STP or TCP) also create output, which is used as an input in another process (like TMP or OTP). Some of the processes (for example TMP) are also the owners of the other processes, such as TPP or TCP. The relationships between different processes and model layers are illustrated in Figure 1.

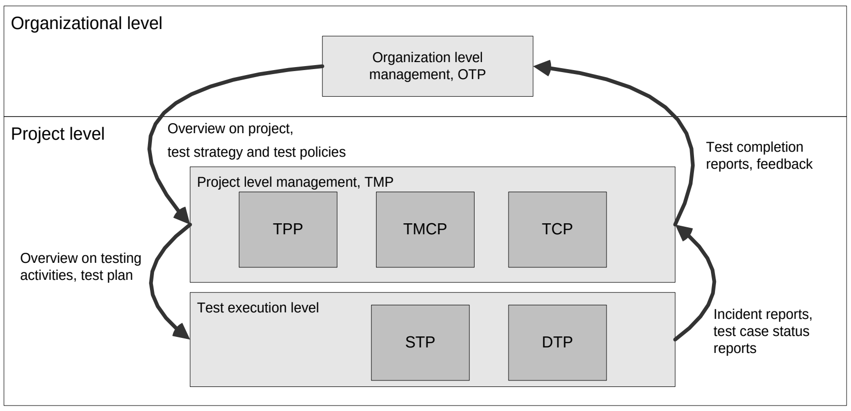

Organizational level
Organization level management, OTP
Project level
Overview on project, test strategy and test policies
Test completion reports, feedback
Project level management, TMP
TPP
TMCP
TCP
Overview on testing activities, test plan
Test execution level
STP
DTP
Incident reports, test case status reports

Fig. 1. ISO/IEC 29119 Test process model in a nutshell.

3.2 Maturity levels for framework

The Test Improvement Model (TIM) [5] is a maturity model based on SEI’s Capability Maturity Model (CMM) [see 7], focusing on test process improvement. The model applies key areas (KA), which are similar to CMM’s key process areas (KPA), assessing important process factors in five levels of maturity. Unlike some improvement models such as TMMi [6], TIM encourages process improvements even in a single key area, as they are assessed separately and are rather independent. Even though it is possible to focus just on one or two process areas, TIM encourages balanced improvement. In a TIM model, level requirements are detailed for all key areas, which are *organization, planning and tracking, test cases, testware* and *reviews*. Even though the actual definitions for each level in different key areas vary, they can be generally defined as follows:

- *Level 0, Initial*: This represents the level of activity where the baseline is not measured or is non-measurable, meaning that the assessed activity does not exist in the organization, is not measurable, or it has not been measured.
- *Level 1, Baseline*: The organization has documented, generally agreed methods of doing testing, and it does basic testing functions, having dedicated people and resources for doing the testing tasks.
- *Level 2, Cost-effectiveness*: The organization has systematic efforts to become cost-effective and increase efficiency of product problem detection.
- *Level 3, Risk-lowering*: The organization is prepared to act on undesired effects. The organization applies measurements and is capable of early involvement and preventative actions to lower the risks of the project.
- *Level 4, Optimizing*: The organization is able to do Quality Assurance and the testing is fully integrated to the development project. Testing activities are continuously maintained and improved based on quality policies, needs and metrics.

In the TIM model, the key areas are assessed separately from each other, so that the organization has a better understanding on what test process areas mostly need improvements. Furthermore, the key areas of TIM maturity model are similar to ISO/IEC 29119 processes; the organization is conceptually close to organizational management process (OTP), planning and tracking to test management process (TMP) and TMCP, test cases to test plan process (TPP), testware to STP and DTP, and reviews to TCP. In this sense, TIM-based maturity levels seem feasible for the purposes of the framework.

4 Pilot study

In this section we introduce the pilot study, which was used to assess the feasibility of the conceptual framework. This section is divided into three parts; in first part, the data collection method and the case organizations are introduced, and the framework is used to develop profiles for each of the organizations. In part two, the case profiles are examined to derive process improvement objectives, and in part three, these objectives are discussed and assessed based on their feasibility and feedback provided by some of the profiled organizations.

4.1 Data collection and analysis

The framework was tested with four different case organizations, on which sufficient data existed to enable the analysis of test process activities. These case organizations were selected from our population of organization units [12], on which empirical data was collected for our previous studies [for example 13, 14] in test process improvement. These organization units represent different sizes [10] and types of professional software development:

- MobileSoft is a large, internationally operating software company, producing embedded software for mobile platforms.
- SoftPortals is a small-sized nationally operating company, producing service solutions for customer organizations.
- DesignSoft is a large, internationally operating software company, producing a software product used for computer-assisted design.
- GridSystems is a medium-sized internationally operating company, producing embedded software for electrical networks and heavy industry.

The interview data was collected in four interview rounds between fall 2008 and spring 2010. In four interviews, the representatives from each case organization were interviewed for several testing-related themes. In all organizations, the interviewees were selected from the same organization unit [12], meaning that the interviewees were working on the same development projects. A summary of these interview rounds is listed in Table 1, and the original questionnaires, with the info package for the interviewees, can be accessed at *http://www2.it.lut.fi/ project/MASTO/*.

Table 1. Data collection rounds.

Round: Type of interview	**Interviewee role in the organization**	**Interview themes**
1st round: Semi-structured interview	Software designer or people responsible for software design and architecture.	Design and development methods, Testing strategy and -methods, Agile methods, Standards, Outsourcing, Perceived quality
2nd round: Structured survey, additional semi-structured questions	Manager, test manager or project leader responsible for development project or test process of a product.	Test processes and tools, Customer participation, Quality and Customer, Software Quality, Testing methods and -resources
3rd round: Semi-structured interview	Tester or people responsible for doing testing in the development project.	Testing methods, Testing strategy and –resources, Agile methods, Standards, Outsourcing, Test automation and –services, Test tools, Perceived quality Customer in testing
4th round: Semi-structured interview	Manager, test manager or project leader responsible for development project or test process of a product.	Test policies, Test strategies, Test plans, Testing work, Software architecture, Delivery models, New software development concepts

The amount of data collected from the case organizations allowed the research to focus on observing how the different standard processes are implemented in these organizations. Based on the empirical data it appeared that all the standard-defined test processes seemed to at least exist, so technically the assessment was feasible. The observations, made on each standard-defined process, are summarized in Table 2.

For the pilot study with the framework, the ISO/IEC 29119 processes were combined with the TIM-maturity scale. The used scale was based on the general definitions of TIM levels, and only few changes were done to the general definitions. For example, the level 1 requirement for the documented test process was eased to accept also verbally agreed testing practices if they were done in a systematic manner. In addition, as the standard steers the test process activities towards continuous development and measurement, the level 4 was used to denote a self-optimizing process, which is in conformance with the standard. Overall, the levels for assessing the organizations were applied with the following rules:

- *Level 0, Initial*: The organization does not have defined methods for this activity.
- *Level 1, Baseline*: The organization does have documented or at least generally agreed guidelines for these process activities, the process is systematically done.
- *Level 2, Cost-effectiveness*: The organization tries to systematically promote cost-effectiveness or increase the efficiency of the process activities.
- *Level 3, Risk-lowering*: The organization has metrics or other methods to enable organization to do risk-lowering and preventative actions in process activities.
- *Level 4, Optimizing*: The organization has activities that aim to optimize the process; activities are done in a manner that is conceptually similar to the standard.

The resulting profiles of case organizations are presented in Figure 2. Based on the observations made in organizations, a draft for practical descriptions indicating the maturity levels of different test processes was also compiled. This draft and an example of full case profile can be accessed at *http://www2.it.lut.fi/project/MASTO/*.

Table 2: Process activity observations from case organizations

Process	**MobileSoft**	**GridSystems**	**SoftPortals**	**DesignSoft**
OTP	-Applies quality model, test strategy and policy exist. -Organizational management tends to underestimate testing.	-Test process defined as a "guideline", vague documentation on the topic.	-Test process defined as a "guideline", part of quality system.	-Policy and Strategy exist, high abstract level. -Organization applies quality model.
TMP	-Management decides what test activities are done. -Management defined, but passive.	-Management sets focus of test cases. -Test management can influence on release schedules.	-Management defined, but passive. -Roles and responsibilities in project level clear.	-Management lays test plan, weekly meetings. -Test management can influence on release schedules.
TPP	-Test plans are tailored to projects; checklists for required tests. -Plan is kept updated, new cases added based on found issues.	-Test plan based on found issues in previous projects.	-Plan is used as an overview for test objectives, little content updates during the project.	-Test plan follows abstract design, design sometimes updated.
TMCP	-Checklists to ensure test coverage -Error database to overview case completion. -No separate metrics.	-Daily SCRUM meetings. -Case status evaluation at the organizational level. -Test focus slipping.	-Customer-required functionalities are inspected closely. -Use case status, unit test coverage used as metrics.	-Problems with follow-up on found issues. -Weekly meetings and code reviews. -Use case status and used hours as metrics.
TCP	-Test process is evaluated after every project. -Continuous development.	-Test completion reports used in new plans. -Effort to increase usage or error reports.	-Test completion reports are done, but little effect to testing practices.	-Test process is evaluated after projects, some effect on future testing.
STP	-The amount of available test tools restricts testing. -Effort to increase amount of testers, tools.	-Amount of test resources sufficient for tasks. -New test cases created according to focus areas defined by management.	-The amount of available resources sometimes restricts testing work.	-Test resources, environments, are documented. -Amount of test resources sufficient for tasks.
DTP	-ISEB-certified testers, tools selected based on recommendations. -No technical limitations caused by testing tools. -Tests follow test plan closely.	-Tests follow test plan closely. -Large amounts of automation -Effort to increase amount of test resources like personnel.	-Some test cases are designed but not implemented. -Test cases are strictly followed and reported forward.	-Test plan and exploratory testing, results reported. -Sufficient tools, effort to automate. -Effort to introduce ISEB-certification.

4.2 Developed organizational profiles

The observations made from organizations made it possible to assess the maturity levels of each standard-defined process separately. This resulted in four profiles, each defining one case (illustrated in Figure 2). Based on these profiles, it was possible to generate two results, the general maturity level of the test process in case organization and the performance level of the processes in the organization. The general maturity level would indicate the average test process maturity and how closely the organization resembled the ISO/IEC 29119 test process model. The performance levels for different processes could be used to prioritize process development to focus, and correct, problems in the individual test activities.

In MobileSoft, the general results indicated that the test processes were generally in a good order. The most obvious difficulties in the organization were the organizational management, which commonly underestimated the test process resource needs. The test completion reports were sometimes not used and some testers complained that there were periodical resource shortages on testing tools like test platforms.

In DesignSoft, the profile also indicated rather mature test process, having only a few issues that should be attended. Based on the observations, in DesignSoft the biggest issue was in follow-up actions if problems were observed during testing. The problem was that discovered issues did not seem to affect the test plan, and follow-up on resolving newfound issues was left to the tester who originally found the problem.

In SoftPortals the test process is less mature than in the other case organizations. The profile suggests that the test management in projects should be improved to enhance the overall testing. In this organization, the management takes a rather passive role on the test process, leaving much of the test planning and control to the developers and testers. In addition, some interviewees complained that the projects usually did not have enough test resources, like dedicated testers or time.

In GridSystems, the test process was divided between extremely well-organized activities in the project level, and the organizational activities which needed improvement. The testing infrastructure employed heavily into test automation,

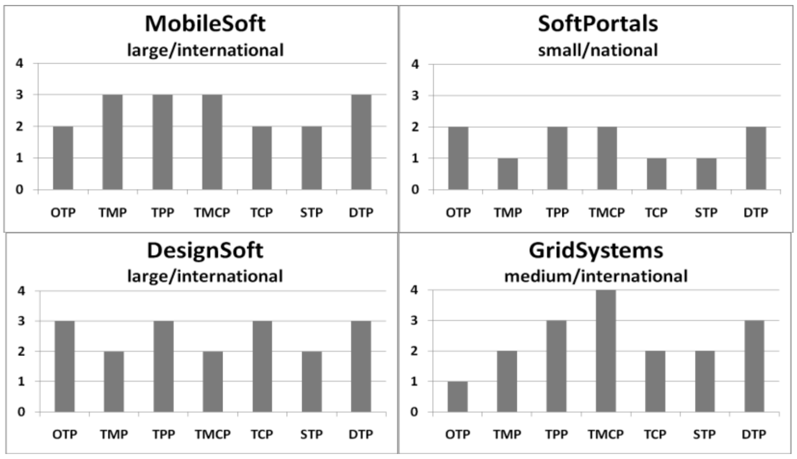


Fig.2. Assessment results for case organizations using experimental levels

having daily builds and Scrum-meetings. This enabled the organization to address test process issues within the same day the problems were observed. For these activities, the organization was considered to fully implement the concepts of the testing standard. However, a large issue was the organizational level management, as the OTP was only a distant decision maker, which provided only some vague guidelines.

4.3 Assessing framework results

The application of TIM maturity levels to the ISO/IEC 29119 standard was successful in a sense that the models are similar; the key factors from TIM and the test processes from ISO/IEC 29119 seem to map consistently to each other. We observed the case organizations using this framework and created an organizational profile based on the observed activities.

In the cases GridSystems and DesignSoft, the framework indicated clear process improvement objectives. In GridSystems, it seemed rather clear that the organization-level management needed to start paying attention to the test process activities. In DesignSoft, the framework indicated the need for developing test completion reports and follow-up activities on found issues during testing. In MobileSoft and SoftPortals the results based on the framework were more open to interpretation, but they did indicate some considerations for development objectives.

According to the profile, the case SoftPortals was the least mature organization in this study. However, based on the interviews, the organization itself does not consider their test process to be faulty or in need of major changes. Their test process seems to aim at cost-effectiveness and efficient application of the existing resources. In this sense, the framework indicates that the management should take a more active role in the process, and that test completion reports are not used efficiently. Should the organization start pursuing the risk-lowering strategy or optimizing test process, they are in need of a larger process improvement process.

In MobileSoft, the framework highlights some objectives for a better risk-lowering strategy. In this case organization, the test process in general is mature, and of the case organizations it is the best candidate to achieve the conformance with the ISO/IEC 29119 standard. The organization could capitalize the test completion reports and experience gained from completed projects more efficiently and the organizational management could be more active in the test process. The organization also follows an internal quality management system, which undergoes an assessment every few years. By making some improvements, it could be expected that they would reach the conformance with relatively minor changes to their test process.

In theory the framework seems plausible, but obviously the analysis based on pre-existing data offers only indications. To evaluate the feasibility of the framework further, three of the profiled case organizations were interviewed to assess the framework results. In the fourth case, GridSystems, the organization had changed so much that this further assessment was not possible. The overall attitude towards the framework and profiling results was positive, although further development needs were identified. Some of the feedback results are summarized in Table 4, where the sign "+ +" denotes that the organization was very positive towards the aspect of the framework, while "– –" denotes very negative feedback.

Table 4: Profiling feedback from the case organizations

	MobileSoft	DesignSoft	SoftPortals
Suitability of the framework	+; The applied approach is generally feasible.	++; Practical approach on quick and easy assessment of the level of different testing tasks.	+; Levels are too universal, but model itself seems to cover everything needed.
Suitability of the assessment levels	– –; In large organization, the levels overlap, unnecessary processes for some organizations.	+; Usable, although some processes do not need to be better than cost-effective.	– ; Levels in general are OK but the definitions should be less ambiguous.
Accuracy of the profile	– ; The profile should be more detailed.	+; The profile was accurate enough, although with some differences.	++; The profile represents the organization quite well.
Accuracy of the results	+; This type of feedback is always good for bringing out new ideas.	+; Results seemed usable.	++; Results same or similar to the internal discussions.
Framework development proposals	The assessment unit type and size should be clearly defined.	More definite descriptions for each framework level to reduce the overlap.	The assessment needs practical examples and more measurements.
Best profiler	An outsider from a third party, internal review is not accurate.	At least two manager-level employees; can be used internally.	A quality manager with a handpicked group of people, usable internally.

Based on the feedback, the framework is considered to be feasible with some criticism. In cases MobileSoft and SoftPortals the criticism focused on the ambiguity of the level definitions. The levels should have detailed metrics, or at least offer examples on what types of activities denote certain levels. MobileSoft also criticized the number of processes; in a software organization which is a part of a larger business unit, some of the activities were considered trivial. DesignSoft voiced a concern over the model levels; lowering risks may not always be a better objective than cost-effectiveness. As for the self-assessment, DesignSoft and SoftPortals considered the framework to be usable as a self-assessment tool, while MobileSoft voiced a concern over the accuracy of the self-assessment in general.

5 Discussion

Creation of a framework which combines maturity levels from one model and processes of an international standard is obviously open for criticism and discussion over its validity. It can be argued that the general requirements for any relevant construct should include at least that it is acceptable to the software development community and that it is based on agreed software engineering principles and practices [15]. The objective of the framework was to compare the existing test process against the ISO/IEC 29119 standard model. Based on the results of using the framework it was also possible to derive process improvement objectives, which would direct the organization towards practices defined in the standard. For this ability, an existing maturity model was fitted to the standard-defined processes. Both of the applied models, ISO/IEC 29119 and TIM, are well-known software engineering models, so theoretically the foundation for our framework should be sound.

The validity issues for developing frameworks have been addressed in several similar studies [8,11,15]. For example, Jung [8] developed a test process maturity model based on internal need, and validated the results via a case study and a survey. Similarly, with the MTPF-framework developed by Karlström et al. [11], the initial model was designed based on observations in real-life organizations, and further elaborated and validated with surveys and an empirical case study. If we compare these approaches to our framework, it is plausible to argue that the current results should be sufficient for a proof of concept: the results indicate that the framework could be developed into a usable tool, but obviously the framework needs further validation. The next obvious step is to address the observed difficulties and enhancement proposals. In general, more detailed qualitative studies and additional data are needed.

6 Conclusions

In this paper we presented a self-assessment framework that combines the ISO/IEC 29119 test process standard [4] with maturity levels from Test Improvement Model (TIM) [5]. The objective was to create a concept-level self-assessment tool to find development objectives for test process improvement and achieving conformance with the upcoming standard. The current limitation of the standard is that it does not offer support for adopting the standard in real-life software organizations, so the framework was defined to enable a maturity level-type self-assessment.

The self-assessment framework was developed and tested with pre-existing data from four organizations, and the results were confirmed with additional interviews with the profiled organizations. The results indicate that the framework could be developed to a useful self-assessment tool for organizations to compare their existing test processes against the standard and find process areas that could be improved. However, areas that require further development in the proof-of-concept version of the framework were also pointed out, and several concrete development proposals from case organizations were collected.

Overall, the results indicate that the framework proposed in this paper could become a feasible tool for defining process improvement objectives which promote the practices defined in the upcoming ISO/IEC 29119 standard. However, the current framework obviously needs further development and studies for validity. One possibility for the next logical step would be to conduct a qualitative study by applying the self-assessment framework in real-life organizations, and studying the applicability or relevance of the results the framework produces, when compared with other existing process development methods.

Acknowledgements

This study was partially supported by the ESPA-project (http://www.soberit.hut.fi/espa), funded by the Finnish Funding Agency for Technology and Innovation and by the companies mentioned in the project web site.

References

1. Myers, G.J.: The Art of Software Testing, 2nd edition, John Wiley & Sons, Inc., New Jersey, USA, (2004).
2. Kit, E.: Software Testing in the Real World: Improving the Process, Addison-Wesley, Reading, MA, USA, (1995).
3. Huang, L. and Boehm, B.: How Much Software Quality Investment Is Enough: A Value-Based Approach, IEEE Software, Vol. 23(5), pp. 88-95, doi: 10.1109/MS.2006.127, (2006).
4. ISO/IEC: ISO/IEC WD 29119-2, Software and Systems Engineering - Software Testing - Part 2: Test Process (2010).
5. Ericson, T., Subotic, A., Ursing. S.: TIM - A Test Improvement Model, Software Testing, Verification & Reliability (STVR), vol. 7 (4), pp. 229-246, John Wiley & Sons, Inc., (1997).
6. Test Maturity Model integration (TMMi), Version 3.1, TMMi Foundation, Ireland (2010).
7. CMMi Product Team: "CMMI for Development, Version 1.3", Software Engineering Institute, Carnegie Mellon University, (2010), URL: http://www.sei.cmu.edu/cmmi/.
8. Jung, E.: A Test Process Improvement Model for Embedded Software Developments, Proc. Of the 9th Internatinal Conference on Quality Software, 24.-25.8.2009, Jeju, South Korea, (2009).
9. Oh, H., Choi, B., Han, H., Wong, W.E.: Optimizing Test Process Action Plans by Blending Testing Maturity Model and Design of Experiments, Proc. of the 8th International Conference on Quality Software, pp. 57-66, doi: 10.1109/QSIC.2008.19 12.-13.8.2008, Oxford, UK, (2008).
10. EU: SME Definition, European Commission, (2003), Available at : http://ec.europa.eu/enterprise/policies/sme/facts-figures-analysis/sme-definition/index_en.htm.
11. Karlström, D., Runeson, P., Nordén, S. : A minimal test practice framework for emerging software organizations, Software Testing, Verification and Reliability (STVR), Vol. 15(3), pp. 145-166, doi: 10.1002/stvr.317, John Wiley & Sons Inc., (2005).
12. ISO/IEC: ISO/IEC 15504-1, Information Technology - Process Assessment - Part 1: Concepts and Vocabulary (2002).
13. Kasurinen, J., Taipale, O., Smolander, K. : Software Test Automation in Practice: Empirical Observations, Advances in Software Engineering, Special Issue on Software Test Automation, Hindawi Publishing Co. DOI: 10.1155/2010/620836, (2010).
14. Kasurinen, J., Taipale, O., Smolander, K. : Test Case Selection and Prioritization: Risk-based or Design-based?, Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 16.-17.9.2010, Bolzano-Bozen, Italy, doi: 10.1145/1852786.1852800, (2010).
15. Burnstein, I., Suwanassart, T., Carlson, R.: Developing a testing maturity model for software test process evaluation and improvement, International Test Conference 1996 (ITC'96), doi: /10.1109/TEST.1996.557106, (1996).

Appendix II: Survey instrument

MASTO Survey and themed questions, round 2; test or project managers

1. Interview

Date

Place

Interviewer

Interview started

2. Respondents

Name Occupation Responsible for development/testing/both

3. Company

Name

Organizational unit (OU)

Industry sector

4. Personnel, methods, and automation

Number of employees in the whole company? (text field)

Number of SW developers and testers in the OU? (text field)

Percentage of automation in testing? (text field)

Percentage of agile (reactive, iterative) vs plan driven methods in projects? (text field)

Percentage of existing testers vs resource need? (text field)

5. Please, estimate the distribution of the turnover in your OU.

Percentage of the turnover 0-20,21-40, 41-60,61-80,81-100%

- Product: Customized product (based on a product kernel)
- Product: Uniform product kernel in all deliveries
- Product: Product family composed of distinct components
- Product: Standardized online service product (e.g. product/service prov.)

- Service: Training and consulting
- Service: Subcontracting
- Service: System integration
- Service: Installation service
- Service: Self service (e.g. service/service provider)
- Other, specify

6. Please, estimate how the following claims describe your software development.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- We like to transfer knowledge more by face-to-face conversation than by documents as the primary method of knowledge transfer.
- Progress of the software is more important than thorough documentation.
- Business people and developers work daily together in the projects.
- Our process is able to cope with late changes in requirements, design, and technical platform.
- We prefer more individuals, collaboration, and interaction than processes and tools.

7. Faults in your products can cause (please, select all suitable points)

Irritation and dissatisfaction

Disturbance in the normal operation of the organization or a person

Remarkable economical losses

Interruption in the normal operation of the organization or a person

Loss of human life

Other, specify (text field)

8. Please, estimate following claims concerning your software testing.

When the claim is not applicable leave the scale empty.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Our software correctly implements a specific function. We are building the product right (human examination).

Our software is built traceable to customer requirements. We are building the right product.

Our formal inspections are ok (document to be inspected).

We go through checklists (req., func., tech., code).

We keep code reviews.

Our unit testing (modules or procedures) is excellent.

Our integration testing (multiple components together) is excellent.

Our usability testing (adapt software to users' work styles) is excellent.

Our function testing (detect discrepancies between a program's functional specification and its actual behavior) is excellent.

Our system testing (system does not meet requirements specification) is excellent.

Our acceptance testing (users run the system in production) is excellent.

We keep our testing schedules.

Last testing phases are kept regardless of the project deadline.

We allocate enough testing time.

9. Please, estimate following claims.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Quality is built in development.

Quality is built in testing.

Our test policy is excellent (principles, approach, and high-level objectives).

Our test strategy is excellent (a reusable set of guidelines for all projects).

Our test management is excellent (strategizing, planning, monitoring, control, and reporting of testing).

Our test execution is excellent (testing within a particular level of testing (e.g. unit, integration, system or acceptance) and/or type of testing (e.g. performance testing, security testing, functional testing).

10. Do you follow a systematic method or process in the software testing (e.g. TPI, or standard, or your own specified process)?

No

To a certain extent; which one (text field)

Yes, which one (text field)

11. The available testing tools, if any

Tool name	Description/Experiences/Recommendations	Inhouse/Vendor

(text field)

12. "Please, estimate your most important customer's participation during specification phase of the development."

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Our most important customer is a member of the project team and responsible for the definition of the system.

Our most important customer takes part in the project management schedules and progress reports for the development of the system.

Our most important customer develops and evaluates the budget for the system.

13. "Please, estimate your most important customer's participation during design phase of the development. "

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Our most important customer is a member of the project team for user interface design.

We develop a prototype for our most important customer.

Our most important customer defines system controls and security procedures.

Our most important customer defines and reviews technical designs.

14. "Please, estimate your most important customer's participation during testing phase."

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Our most important customer develops test specifications.

Our most important customer evaluates test data specifications developed by us.

Our most important customer reviews results of system test done by us.

Our most important customer conducts the system tests.

15. Please, estimate your most important customer's participation in general control.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Needed new features are paid by our most important customer.

Our most important customer reviews project management schedules and progress reports made available by us.

Our most important customer provides domain training to us.

Our most important customer's employees are evaluated by their own management in our collaboration projects.

16. Please, estimate the following claims.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Our most important customer has experience on the area of business.

Our most important customer has power on the area of business.

Our most important customer has strict contractual agreements.

Our most important customer has requests and suggestions.

Our most important customer co-operates and communicates excellently with us.

17. Please, estimate the following claims.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Our most important customer is concerned about our welfare and best interests.

Our most important customer considers how their decisions and actions affect us.

We trust our most important customer.

Our most important customer trusts us.

18. Do you have a quality system certificate or a capability-maturity classification (e.g. CMM, SPICE, ISO-9001)?

No

Yes; Which one (text field)

19. Please, estimate following claims concerning quality attributes of your software.

When the quality attribute is not applicable leave the scale empty.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

The functional stability is excellent. Our software is suitable for functions it is developed for (appropriateness).

The reliability is excellent. The availability, fault tolerance, and recoverability of our software are excellent.

The performance efficiency is excellent. Our software consumes a reasonable amount of resources and time.

The operability is excellent. Our software is useful and usable according to the users (ease of use).

The security is excellent. The security issues (malicious access, use, modification, destruction, or disclosure) have been taken into account.

The compatibility is excellent. Our software is compatible with relevant software or components.

The maintainability is excellent (modifications and changes after delivery).

The transferability is excellent: Our software can be transferred and installed to another platforms.

20 Please, estimate following claims related to your software.

Scale 1= Fully disagree, 3=neutral, 5= fully agree

We have identified the most important quality attributes

We have prioritized the most important quality attributes

We have documented the most important quality attributes

We have communicated the most important quality attributes within our OU using some other way than documentation.

We follow regularly through measurement the achievement of the most important quality attributes.

21. How many percent of the development effort is spent on testing? (text field)

22. Please, estimate following claims concerning problems.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Complicated testing tools cause test configuration errors.

Commercial testing tools do not offer enough hardware support.

It is difficult to automate testing because of low reuse and high price.

Insufficient communication slows the bug-fixing and causes misunderstanding between testers and developers.

Feature development in the late phases of the product development shortens testing schedule.

Testing personnel do not have expertise in certain testing applications.

Existing testing environments restrict testing.

23. Please, estimate following claims concerning enhancement proposals?

Scale: 1=fully disagree, 3=neutral, 5=fully agree

Fault database hepls in observing testing process.

Test report automation decreases testers' work load.

New features are not allowed after a set deadline to help test planning.

Test results should be open for all process groups.

Products should be designed to promote testability.

Testing strategy helps in avoiding unnecessary testing.

We should have dedicated testers.

Development of testing environment makes testing more efficient.

24. Name and explain the three most significant factors how customer affects software quality in projects (in descending order). (text field)

25. Name and explain the three most efficient tools or methods of test automation (in descending order). (text field)

26. Name and explain most important advantages and disadvantages of outsourcing in development and testing. (text field)

Future research and reporting

Please send us the summary report of the research (yes/no)

Contact information

Name Occupation Contact information (e-mail)

(text field)

Comments

(text field)

Interview ended (text field)

Appendix III: Theme-based questions for the interviews

MASTO-project themed questions, round 1; Designers

Topic 1: Design- and development methods

1.1 Which types of development methods for software do you use? Do you apply agile methods? How?/Why not?

1.2 Which is the criticality level for your products? Does it fluctuate? If yes, does it affect the development method for the product?

Topic 2: Testing strategy and -resources

2.1 How does your work relate to testing?

2.2 How do you decide, which test cases are selected? How, in your experience, is this strategy working?

2.3 What part of the testing process you would like to develop? Why?

2.4 Does the product criticality affect the testing strategy? How?[4]

2.5 Does the testing process have sufficient resources? If not, why? What would you do to address this issue?

Topic 3: Agile methods

Asked only if agile methods are applied

3.1 What kind of experiences do you have on the applicability/operability of agile methods?

3.2 Does the application of agile methods affect the component quality or reusability? How?

3.3 Does the agile methods affect the need for testing process resources? How about timetables? How?

[4] Asked if the criticality fluctuates

Topic 4: Standards

4.1 Do you follow any standards in your software process? If yes, what? Which kind of experiences do you have on effects of software standards to the process or product?

- ISO/IEC 29119

4.2 Do you monitor the effectiveness or quality of your testing process? If yes, how? If no, why do you think that it is not followed?

- Which monitoring methods?
- How about on outsourced modules?

Topic 5A: Outsourcing

5.1 Which kind of knowledge is needed to test your product efficiently? How can this knowledge be obtained?

5.2 Do you obtain testing services or program components from outside suppliers? What services/components?/Why not?

Topic 5B: Outsourcing, continued

Asked only if company has outsourced components or services

5.3 Does your production method support outsourced testing services? Why? How about with critical software?

5.4 Does your production method support outsourced/ 3rd party components? Why?

5.5 Does the outsourcing affect the testing strategy? Why?

Topic 6: Testing automation, -services and tools

6.1 Do you use automation in testing? If yes, to which operations it is used? If not, why?

6.2 What sort of experiences do you have on testing automation and in applying automation to testing process?

6.3 Have you found or used testing services or –products from the Internet? If yes, then what kind? What services would you like to find or use from the Internet? Why?

6.4 Are there any testing services or -tools that you would like to have besides those already in use? Why?

Topic 7: Quality and supplier-customer relationships

7.1 How do you define quality, in terms of what quality aspects are important to you? How does this reflect to the development and testing processes? ISO 25010:

- Functionality
- Reliability
- Efficiency
- Usability
- Security
- Compatibility
- Maintainability
- Transferrability

7.2 Does the product criticality affect the quality definition? How? Why?

7.3 Does the outsourcing/3rd party components affect the quality? How? Why?

7.4 How does the customer participation affect the quality?

- … with large size difference between customer and supplier?
- … with trust between customer and supplier?
- … with customer satisfaction?

Topic 8: Meta and other

8.1 To which research area would you focus on in testing?
-Testing policy
-Testing strategy
-Test management
-Test activity
8.2 Is there anything relevant that you feel that wasn't asked or said?

MASTO -project themed questions 3; Testers

Topic 1: Testing methods

1.1 What testing methods or –phases do you apply? (unit, integration, usability, alpha/beta etc.)

1.2 Does the product purpose or criticality affect the testing? Do the testing methods fluctuate between projects? If yes, then how? If no, then should it? Why? (Explain criticality)

Topic 2: Testing strategy and -resources

2.1 How does your work relate to testing?

2.2 How do you decide, which test cases are selected? How, in your experience, is this strategy working?

2.3: Test documentation

- In how fine details your test cases/plans are documented?
- What kind of documentation is the most practical or important to you as a tester?
- Do you do explorative testing?

2.4 Are the testing requirements able to affect the product timetable? How? /Why do you think that it is not?

2.5 Does the testing process have sufficient resources? If not, why? What would you do to address this issue?

2.6 Would like to develop some particular part of the testing process? How/Why?

Topic 3: Testing and Agile methods

3.1 Are agile methods used in your company?Does it affect the testing strategy? How about timetables?

3.2 Does the application of agile methods affect the quality of product or components? How about resource need?

Topic 4: Standards

4.1 Do you follow any standards in your software process? If yes, what? Which kind of experiences do you have on effects of software standards to the process or product?

4.2 Do you monitor the effectiveness or quality of your testing process? Which monitoring methods do you use? If no, why do you think that it is not followed?

- How about on outsourced modules?

Topic 5A: Outsourcing

5.1 Which kind of knowledge is needed to test your product efficiently? How can this knowledge be obtained?

5.2 Do you obtain testing services or program components from outside suppliers? What services/components?/Why not?

Topic 5B: Outsourcing, continued

Asked only if company has outsourced components or services

5.3 Does your production method support outsourced testing services? Why? How about with critical software?

5.4 Does the outsourcing affect the testing strategy? Why? How about quality?

Topic 6: Testing automation

6.1 Do you use automation in testing? If yes, to which operations it is used? If not, why?

6.2 What sort of experiences do you have on testing automation and in applying automation to testing process?

6.3. How large is the portion of manually done software testing? How does it reflect to the product quality?

Topic 7: Testing tools

7.1 Do you use software tools especially made for testing? If yes, then what kind?

- Your opinion regarding these tools.

7.2 Are your tools vendor- or in-house-products? Why do you think this is this way?

- Your opinion regarding quality and efficiency of vendor tools.
- Your opinion regarding quality and efficiency of in-house tools.

7.3 Have you found or used testing services or –products from the Internet? If yes, then what kind? What services would you like to find or use from the Internet? Why?

7.4 Are there any testing services or -tools that you would like to have besides those already in use? Why?

Topic 8: Quality

8.1 Do you know which the quality definitions for the product under testing are? What are they? How does this reflect to the testing process? If not, how would you define them? ISO 25010:

- Functionality
- Reliability
- Efficiency
- Usability
- Security
- Compatibility
- Maintainability
- Transferrability

8.2 Does the product criticality affect the quality definition? How? Why?

Topic 9: Customer in the project

9.1 How does the customer participation affect the testing process? Can the customer affect the test planning or used test case selection?

- … with large size difference between customer and supplier?
- … with trust between customer and supplier?

Topic 10: Meta and other

10.1 To which research area would you focus on in testing?
-Testing policy
-Testing strategy
-Test management
-Test activity

10.2 Is there anything relevant that you feel that wasn't asked or said?

MASTO -project themed questions 4; Test managers

Topic 1: Test Policy

1.1 Does your organisation have a test policy or something resembling it? If yes, what does it define? Does it work? Why?
1.2 If no, does your organisation apply same or similar test process in all projects? Would you think that such a document could be defined in your organisation? Why/Why not?

Topic 2: Test Strategy

2.1 Does your organisation have a defined test strategy or something resembling it?
2.1.1 If yes, what does it define? In your opinion, is it useful?
2.1.2 If yes, is it updated or assessed for change requirements systematically or ”if needed”?
2.1.3 If no, does your test process apply same or similar phases in all software projects? Would you think that test strategy, as described earlier, could be defined based on your test process? Why/Why not?
2.2 Name three most effective testing practices (e.g. explorative testing, code reviews, glass-box testing etc). Why are they effective? Have you defined them in writing? If yes, what details do they include? If no, why?
2.2.1 Would your organisation try out new testing practice from ”best practices”-type instruction manual without prior knowledge regarding this new practice? Why/Why not?

Topic 3: Test Plan

3.1 Do you define test plans for each software project at the design phase?
3.1 If yes, how detailed are they? Do they change during the project?
3.2 If no, would you think that such a plan could be defined in design, or generally before testing is started?
3.2 Who in your organisation defines the test plan (or decides on what is tested)? In your opinion, how much do policies or management affect these decisions? How about resources?
3.3 Do you think that testers should follow definite plans for all test cases? How much details should this plan have?
3.4 Does your organisation do testing-wrap ups such as test completion reports or project post mortems? Do these reports affect how testing is done in the later projects?

Topic 4: Testing

4.1 How does your business orientation (service orientation or product orientation) affect the test process?
Sommerville (1995) classifies software producers into two broad classes according to their software products: producers of generic products and producers of customized products. In a broader sense, business orientation may also mean the basic offer addressed by an organisation to its customers e.g. for an independent software testing provider the basic offer refers to the testing services offered to customers or development services in the case of a software development service provider.
4.1.1 Where do your software and testing requirements originate from?
4.1.2 At which organisational test process level (policy, strategy or plan) are the requirements considered?
4.2 How do the customers/end users affect your test process? Which organisational test process level (policy, strategy or plan) is most affected?
4.3 What are the current unmet customer/end user needs? In your opinion, why do you think they have not been met? How do you plan to fulfil them? Does your test process pay attention to this?
4.4 How do you measure and optimize the testing progress within your organisation? Is it effective? If yes, how? If not, do you have any improvement propositions?
4.5 Does the described ISO/IEC 29119 software testing standard meet your needs? What is missing?

Topic 5: Software architecture and delivery models

5.1 Please, describe your software architecture or software delivery model (e.g. distributed, client-server, SaaS, cloud computing, service oriented, data base centric, component based, structured etc.) In your opinion, how does it affect testing?
5.1.1 Does it cause any problems to your testing process?
5.1.2 If yes, please, give improvement propositions.
5.2 Has your software architecture or your software delivery model changed during the last years? If yes, how and why? If not, why? Do you have any plans to change it? How does this affect or has affected your testing work?
5.3 Do you think that your test process may be affected by new architectures or new software delivery models e.g. SaaS (Software as a Service), cloud computing or open source technology? If yes, how? If no, why not?
5.3.1 Are you searching for any new testing tools or methods? What benefits do they offer?
5.3.2 Does your work make use of systems that require huge amounts of computing power and virtual data storage? If yes, how do you handle it

now? Would you consider resources presented by cloud computing to meet these needs? If yes, what kind of tools you are using?
5.3.3 Please describe how open source technology has affected testing work in your organisation.

Asked if the organisation has used or is considering new software delivery models:
5.4 Have you considered using cloud or SaaS as a delivery model for any of your applications? Have you dealt with service level agreements (SLAs) or pricing models in cloud based testing? Please comment, how they affect your work?
5.5 How is the test data handled? Where does it come from? Who owns it?
5.6 How does your organisation plan on handling/harmonizing test processes across multiple players? How would this affect your overall test process?

Topic 6: Crowdsourcing: New way of sourcing

6.1 According to our earlier survey, the lack of testing resources was in average 25%. What is the situation now and how do you try to solve the lack of resources if needed?
6.2 Does your organisation use (or plans to use) crowdsourcing as a way to complement its internal testing team? If yes, how does this affect your test process? If no, why not?
6.3 If you are interested in crowdsourcing, please explain the most important advantages and disadvantages of crowdsourcing in testing.

Topic 7: Other aspects

7. Is there something you would like to add to your answers or something regarding testing that you think should be mentioned?

MASTO -project themed questions regarding the self-assessment framework results (See *Publication VII*):

-Overall, what is your opinion regarding the assessment framework? Is something missing, are all of the important testing-related aspects considered in the assessment?

-In your opinion, are the defined maturity levels and their descriptions usable/understandable? If no, why?

-Do you think the profile represents your organisation? If no, why? What should be different?

-Do you think the development suggestions are useful for your organisation? If yes, in your opinion, are the changes possible to implement? If no, why do you think that is?

-In your opinion, would you consider this type of self-assessment feasible approach? If yes, who do you think would be the best assessor for your organisation? If no, why? (Assessor can also be a group of people)