Lappeenranta
University of Technology

Frank Philip Seth

EMPIRICAL STUDIES ON SOFTWARE QUALITY CONSTRUCTION: EXPLORING HUMAN FACTORS AND ORGANIZATIONAL INFLUENCES

Thesis for the degree of Doctor of Science (Technology) to be presented with due permission for public examination and criticism in the Auditorium 1382 at Lappeenranta University of Technology, Lappeenranta, Finland, on 12th of August 2015, at noon.

Acta Universitatis
Lappeenrantaensis 642

Supervisors Professor Kari Smolander
Department of Software Engineering and Information Management
School of Industrial Engineering and Management
Lappeenranta University of Technology
Finland

Associate Professor Erja Mustonen-Ollila
Department of Software Engineering and Information Management
School of Industrial Engineering and Management
Lappeenranta University of Technology
Finland

Associate Professor Ossi Taipale
Department of Software Engineering and Information Management
School of Industrial Engineering and Management
Lappeenranta University of Technology
Finland

Reviewers Prof. Jarmo Ahonen
Department of Computer Science
University of Eastern Finland
Kuopio Campus
Finland

Dr. Petri Kettunen
Department of Computer Science
University of Helsinki
Finland

Opponent Prof. Mika Mäntylä
Department of Information Processing Science
University of Oulu
Finland

ISBN 978-952-265-805-0
ISBN 978-952-265-806-7 (PDF)
ISSN-L 1456-4491
ISSN 1456-4491

Lappeenrannan teknillinen yliopisto
Yliopistopaino 2015

Dedicated to my beloved mother, *Hulda Philip Seth,* who refused to believe in the lies that I would end up becoming a cobbler.

"If life were without mothers,

There would be no you or me.

We all need to take hold, of how precious life really is,

and not take for granted the gift of love and the ability to give.

Thank you to God for creating us all

For giving us our mothers

So we can stand proud and tall.

Humans are no accident, no mistake or error

We sometimes live our lives in fear of this,

Even to the point of terror.

But when the truth is told, and recognised by all,

God does not create rubbish. He knew what he was doing when he created

me and you, but the best gift of all, is giving us loving, caring

Mothers who love us unconditionally through and through.

None of us are perfect, that includes ourselves and family.

Hold onto how precious we are and thank God for our mothers.

Amen " - Julia Hunt

Abstract

Frank Philip Seth
Empirical studies on software quality construction: Exploring human factors and organizational influences
Lappeenranta, 2015
76 p.

Acta Universitatis Lappeenrantaensis 642
Diss. Lappeenranta University of Technology
ISBN 978-952-265-805-0, ISBN 978-952-265-806-7 (PDF)
ISSN-L 1456-4491, ISSN 1456-4491

Software quality has become an important research subject, not only in the Information and Communication Technology spheres, but also in other industries at large where software is applied. Software quality is not a happenstance; it is defined, planned and created into the software product throughout the Software Development Life Cycle.

The research objective of this study is to investigate the roles of human and organizational factors that influence software quality construction. The study employs the Straussian grounded theory. The empirical data has been collected from 13 software companies, and the data includes 40 interviews.

The results of the study suggest that tools, infrastructure and other resources have a positive impact on software quality, but human factors involved in the software development processes will determine the quality of the products developed. On the other hand, methods of development were found to bring little effect on software quality. The research suggests that software quality is an information-intensive process whereby organizational structures, mode of operation, and information flow within the company variably affect software quality. The results also suggest that software development managers influence the productivity of developers and the quality of the software products. Several challenges of software testing that affect software quality are also brought to light. The findings of this research are expected to benefit the academic community and software practitioners by providing an insight into the issues pertaining to software quality construction undertakings.

Keywords: Software quality, software quality construction, requirements, functional requirements, non-functional requirements, grounded theory

“Having come into contact with a civilization which has over-emphasized the freedom of the individual, we are in fact faced with one of the big problems of Africa in the modern world. Our problem is just this: how to get the benefits of European society -- benefits that have been brought about by an organization based upon the individual -- and yet retain African's own structure of society in which the individual is a member of a kind of fellowship.” - Julius Kambarage Nyerere as quoted in the New York Times Magazine on 27 March 1960.

Acknowledgements

I consider it a divine miracle to have had the opportunity to pursue studies at doctoral level. Not many people in Africa have such good fortune, not because of academic failings but because of the many obstacles they face, including poverty and inhibited social development. The funds used for this study were intended for academic staff at the Dar es Salaam University College of Education (DUCE). Despite being a member of the administrative staff, my contribution to the academic team as an assistant part-time lecturer and IT specialist was valued such that I was granted this wonderful opportunity to pursue further education.

The journey to a doctorate is never easy and cannot be completed without the support of many wonderful people. Although I am not able to mention the names of everyone who contributed to my success, I acknowledge and deeply appreciate all your invaluable efforts and support.

I would like to thank my supervisors, Professor Kari Smolander, Associate Professor Erja Mustonen-Ollila and Associate Professor Ossi Taipale, for their consistent support, extensive discussions and thoughtful critique of my work. I express my special gratitude to my principal supervisor, Professor Kari Smolander, for accepting me as a doctoral student in the department and guiding me through the world of software engineering. As a leader, Professor Smolander saw me through many troubles on the path of pursuing doctoral studies. I thank Associate Professor Ossi Taipale for accepting me on the STX project, where I carried out my research work, and for facilitating the travel involved in this study. I deeply appreciate the extensive time we spent discussing in your office. Our discussions led to the creation of new ideas and enrichment of my knowledge of the software engineering discipline. I thank Associate Professor Erja Mustonen-Ollila for her tireless all-around support. Your encouragement, kind assistance and thoughtfulness will never be forgotten.

I sincerely thank the preliminary examiners of this doctoral thesis, Professor Jarmo Ahonen, from the University of Eastern Finland, Kuopio Campus, and Dr. Petri Kettunen, from the University of Helsinki, Finland, for examining the manuscript and giving valuable comments. Your feedback and recommendations helped me improve the final version of this thesis.

For financial support, I would like to thank the World Bank, Science and Technology Higher Education Program (STHEP), administered through the Tanzania Ministry of Education and Vocational Training (MoEVT), and the Software Testing for Intended

Quality Project (STX) at Lappeenranta University of Technology (LUT). I thank Professor S. B. Misana and Professor B. B. Nyichomba for the trust you had in me and allowing me to be granted with the STHEP scholarship without which I would have never started my doctoral studies.

I appreciate the support and assistance of my colleagues at the Department of Software Engineering and Information Management. Special gratitude to my colleague and sister, Dr. Leah Riungu-Kalliosaari. We worked as a team and you played an important role of introducing me to the research field. It was such a blessing having you around at the department. You stretched your hand to help me in so many ways. Special thanks to Ms. Tarja Nikkinen, the department secretary, for her kind and swift support in many administrative issues. Tarja, you have been very important in my studying process. No one will understand this unless he/she has been a stranger in a foreign land and has had to ask about everything, even the meaning of the words on train tickets written in Finnish. Thank you very much, Tarja.

From other departments in LUT, I express my gratitude to Dr. Isambi Sailon Mbalawata, who was a mathematics doctoral candidate at LUT. I would never have met Prof. Kari without the kind efforts made by Dr. Isambi. I would like to thank Peter Jones from LUT Language Center. Peter, you laid a firm foundation for my English academic writing. It was alongside your course "English Clinic for PhD Students" that I wrote my first journal article (*Publication II*), which was published in a leading journal in the field (*Software Quality Journal*, Springer). I thank you for your efforts in polishing my English and kind support in editing two of my publications. I would like to thank Ms. Johanna Jauhiainen, the study secretary at LUT, for her kind support and facilitation of the processing of my thesis when I was in Tanzania. Regardless of my absence, Johanna made sure that I received all the necessary information as appropriate and on time.

There is no enough space to mention all my friends around the world. You have been so important in all the seasons of my life. You shared with me both the sad and joyful moments. Thank you for your prayers and encouragement. My special gratitude to Mr. Paul Seever, I can't forget the bridge you laid in front of me without which I would have never crossed to the doctoral level. I still remember what Mary Seever said about me when I was a teenager. While all people looked at me as a village boy, she saw a leader! Thank you very much for your kind contributions towards my academic success.

My deepest gratitude goes to my mother, Hulda Philip Seth, to whom this entire thesis is dedicated. You encouraged me, prayed for me and advised me. Despite being a widow for 38 years at this date, left with 10 children, a peasant in the village, you never gave up on any of your children. I remember you sold your cow, an important source of income, to be slaughtered so that I could start my secondary education. I

thank you for your sacrifices. I fondly remember my father, Mr. Philip Seth, who passed away in 1977 before I stared pre-school. Your words spoken to me when I was an infant still sound in my ears today, and I see them coming to pass. I know you would be proud of this success, dad. You said how the boy should be treated and my mother kept the promise and now I thank her on your behalf. My sisters and brothers, I thank you for your encouragement and prayers.

My dear wife, Esther F. Philip, my beloved sons, Joshua and Timothy, and my beloved daughter, Faraja, you suffered many things when I was away in Europe and you were in Tanzania. I am truly grateful for your love, patience, understanding and support. I believe it was by the Grace of the Almighty God that we have been able to achieve what we have. Thank you Jesus.

Dar es Salaam, May 2015

Frank Philip Seth

List of publications

I. Seth, F. P., Mustonen-Ollila, E., Taipale, O., and Smolander, K. (2012). "Software Quality Construction: Empirical Study on the Role of Requirements, Stakeholders and Resources", Proceedings of the IEEE 19th Asia-Pacific Software Engineering Conference (APSEC), 4-7 December 2012, Hong Kong, pp. 17-26.

II. Seth, F. P., Mustonen-Ollila, E., Taipale, O., and Smolander, K. (2014). "Software Quality Construction in 11 Companies: An Empirical Study using the Grounded Theory", Software Quality Journal, DOI 10.1007/s11219-014-9246-2.

III. Seth, F. P., Taipale, O., and Smolander, K. (2014). "Organizational and Customer related Challenges of Software Testing", Proceedings of the IEEE 8th International Conference on Research Challenges in Information Science (RCIS), 28-30 May 2014, Marrakesh, Morocco, pp. 1-12.

IV. Seth, F. P., Mustonen-Ollila, and E., Taipale, O. (2014). "The Influence of Management on Software Product Quality: An Empirical Study in Software Developing Companies", Proceedings of the 21st International Conference on European System, Software & Services Process Improvement & Innovation (EuroSPI), 25-27 June 2014, Luxembourg, Volume 425, 2014, pp. 147-158.

V. Seth, F. P., Taipale, O., and Smolander, K. (2014). "Role of Software Product Customer in the Bring Your Own Device (BYOD) Trend: Empirical Observation on Software Quality Construction", Proceedings of the 15th International Conference on Product-Focused Software Process Improvement (PROFES), 10-12 December 2014, Helsinki, Finland, LNCS 8892, pp. 194-208.

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

Symbols and abbreviations

BYOD	Bring Your Own Device
CMU/SEI	Carnegie Mellon University (CUM) and Software Engineering Institute (SEI)
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
EU	European Union
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
ISO/IEC	International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC)
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
NASA	National Aeronautics and Space Administration
R&D	Research and Development
SDLC	Software Development Life Cycle
SME	Small and Medium-sized Enterprises
SPPA	Software Project Planning Associate
SEI	Software Engineering Institute
SUT	System under test

Contents

List of figures

List of tables

1 Introduction

In this era of science and technology, software has become a key player that has increasingly added the value of hardware and services. There has been an increase in software dependence. Software has become a part of the daily life of people in many aspects and is widely applied in areas such as home, offices, transportation, military, agriculture, outer space technology, etc. For example, the demand of software stretches from simple home-based applications, such as toys, games, television sets, etc., to critical systems such as life supporting systems in hospitals, air traffic control systems, etc. Software failures, lack of safety and poor design, have negative consequences to users, infrastructures, businesses, and software developers. Poor quality software may cause loss of life, economical losses and legal implications. Thus, the quality of software is necessary regardless of the criticality of the systems or area of application.

In this thesis, the main goal is to *increase empirical knowledge of the role of humans and software companies in software quality construction*. The scope of the study is limited to activities that are carried out within software developing companies. The focus on 'human' and 'organization' has been motivated by the study of Petre (2010), who claims that despite of development in technology, the result of any activity depends on how well a human can reason and perform the task. Furthermore, Cockburn and Highsmith (2001) argue that if the people (the human factors) in a software development project are good enough, they can use almost any process to accomplish their assignment successfully. These two studies suggest that despite of technology, such as tools, infrastructure, and processes, the human is the key success factor in developing products; and software quality is not an exception (Chotisarn and Prompoon, 2013). For example, Chotisarn and Prompoon (ibid.) argue that the human

is the most important resource in a software development project, and has become the key success factor of software projects in terms of quality. I address the goal of this thesis by providing empirical results that help to explain and understand human and organizational factors pertinent to software quality construction.

The research question of this study is "How is software quality constructed in software developing companies?" A series of empirical studies have been conducted by using qualitative research methods, and following the grounded theory (Strauss and Corbin, 1990). The grounded theory was used in data collection and analysis. The obtained results are presented in *Publications I-V* (Appendix 1).

The contribution of this thesis can be set into two major categories: first, human factors (developers and customers), and second, organizational factors (management, tools, organizational structures, and processes). In human factors, I provide insights into the role of requirements, stakeholders and resources in software development and testing. I further explore the role of the software product customer in the Bring Your Own Device (BYOD) trend, and the impact on software quality. In the organizational factors, I provide insights into the role of management and organizational structures in software development activities, and present issues that influence software product quality.

The thesis consists of two parts, an introduction and five scientific publications as appendix 1 and the data collection tools as appendix 2. The introduction presents the background of the research, research goals, research question, research methods, overview of the publications, and a summary of the contribution of the study. The appendix contains the publications, which present the research results in detail. All the publications are peer-reviewed, and have been published in journals or conference proceedings.

The introduction contains six chapters. Chapter 2 presents the background of the research area and the context of the thesis. Chapter 3 describes the research problem and its shaping, research methods, and research process. Chapter 4 is an overview of the publications. Chapter 5 discusses the contribution, theoretical and practical implications of the research, and the validity and limitations of the study. Finally, Chapter 6 contains a summary and discussion of the contribution of the thesis, including future research topics.

2 Software quality construction

This chapter presents the themes, concepts and definitions forming the background of the research work described in this thesis. The aim of this chapter is to describe the context, and to set the scope of the thesis. The definitions and concepts have been obtained from related research and literature. The chapter presents the background of software quality, definition of software quality, definition of software quality construction, and the roles of humans and software developing companies in the software quality construction process.

2.1 Background of software quality

The history of software quality can be traced back to the 1940's, at the start of programming languages. Since the beginning, writing software has evolved into professional concern with how to improve the quality of software. There were no standards or frameworks guiding programmers to evaluate the quality of their software. The major concern was "correct input for correct output"; anything within a program that produced unexpected results was termed as a bug (i.e. faulty code) (Gokhale et al., 2006; Mudunuri, 2010). According to Mudunuri, in the 1940's till mid 1950's, testing aimed at fixing bugs, and so it was called debugging. Yet, debugging is a common practice even to date (Gokhale et al., 2006). In the 1940's, Grace Murray Hopper, an American computer scientist, popularized the term "debug" (Wexelblat, 1981), referring to fixing problems in computer programs.

As the use of computers and software grew, more software quality concerns were paid attention to. Issues such as maintainability, stability, speed, usability, testability,

security and customer satisfaction, among many other attributes, were thought to be important when creating software. In the 1970's, McCall et al. (1977) described quality characteristics and their attributes. Boehm et al. (1978) published the first hierarchical quality model. Since then, several variations of quality models have been proposed. The models include hierarchical models, meta-model-based models, and implicit quality models (Kitchenham and Pickard, 1987; Kläs et al., 2009). The models have been used to evaluate the quality of software, focusing on particular attributes of software characteristics.

In 1991, the International Organization for Standardization (ISO) developed the software quality standard ISO/IEC 9126-1 (2001), which also introduced the first quality model standard. The ISO/IEC 9126-1:2001 quality model agreed with previous software quality models (McCall et al., 1977; Boehm et al., 1978), with a few variations. The ISO/IEC 9126-1:2001 model was later updated and replaced by the model introduced in the standard ISO/IEC 25010 (2011). The ISO/IEC 25010 (2010) defines software attribute as an "inherent property or characteristic of an entity that can be distinguished quantitatively or qualitatively by human or automated means". According to ISO/IEC 25010 (2010), "software characteristics and subcharacteristics provide consistent terminology for specifying, measuring and evaluating system and software product quality. They also provide a set of quality characteristics against which stated quality requirements can be compared for completeness."

The understanding and definition of quality evolved as more researchers conducted research on the subject. The definitions of software quality have been derived from other fields of study, such as production industry and business (Crosby, 1979; Garvin, 1984). For example, in the 1970's, the definition included "*absence of defects or bugs*" as an indicator of a good software quality. For example, Crosby (1979) defined quality as zero defects. This definition is based on testing process that aims at detecting and fixing bugs, and the software to produce desired outputs. In the 1980's, software quality definitions addressed "*satisfaction of intended and implied customer requirements*". For example, ISO/IEC 9126-1 (2001) defines software quality as the totality of features and characteristics of a product or service that bears on its ability to satisfy specific and implied needs. This new understanding of quality considers software quality beyond fixing bugs, and delivering desired outputs; the quality is evaluated by focusing on how well customers' specific and implied needs are met. The ISO/IEC 25010 (2010) uses the term "requirements" instead of "needs". Literally, requirements are derived from identified needs. Furthermore, the ISO/IEC (25010) and ISO/IEC 25020 (2007) extended the scope of quality model to include the context of user of software products and systems, and suggested the definition for *quality in use* which states "quality in use is the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use" (ISO/IEC

25010, 2010, p. 16; ISO/IEC 25020, 2007, p. 12).

2.2 Definition of software quality and software quality construction

Quality is an abstract term that is difficult to define. Many researchers have tried to define quality and have suggested many definitions. For example, Kitchenham and Plefeeger (1996) use various definitions of quality suggested by Garvin (1984), to bring an insight into software quality. According to Garvin (ibid.), quality is defined according to stakeholders' perspectives, and he suggests five definitions based on those perspectives: *transcendent, product-based, user-based, manufacturing-based* and *value-based* perspectives.

The *transcendent perspective* views quality as something towards which strives as an ideal, but may never be implemented completely (Garvin, 1984). In the *product-based perspective*, quality is viewed as quantifiable and possesses measurable characteristics or attributes. For example, durability or reliability can be measured by Mean Time Between Failures (MTBF) or Mean Time To Repair (MTTR); performance and robustness can be measured through stress testing (i.e. testing that tries to break the system under test (SUT) by overwhelming its resources or by taking resources away from it); recoverability can be measured through load testing (i.e. constantly increasing the load on the system via automated tools) (Gheorghiu, 2005). The *user-based perspective* views quality as an individual matter, and quality is regarded as the satisfaction of user preferences (i.e. perceived quality) (ISO/IEC 25020, 2007; Kitchenham and Plefeeger, 1996). The *manufacture-based perspective* views quality as conformance to requirements or specifications, and any deviation from requirements or specification is regarded as reduction of quality. The *value-base perspective* views quality in terms of costs and prices, as well as a number of other attributes. Kitchenham and Plefeeger (1996) suggest that a customer may decide to buy a quality product, but at an acceptable price.

Although Garvin's (1984) definitions of quality were meant for production industry, the definitions can be applied to software quality as well. Kusters et al. (1997) argue that different views of quality rightfully exist and should not be ignored. Generally, ISO/IEC 25020 (2007) defines quality as "*the ability of a software to satisfy stated and implied needs when the software product is used under specified conditions*". In this definition, the customer need is the basis of the quality. Although software or systems are developed for customers, there are organizational goals. Software projects have objectives beyond customer satisfaction. With this respect, quality should be defined

at a broader perspective to address organizational goals such as increasing market share, building brands, avoiding legal actions and losses, increasing economic gains, etc. For example, National Aeronautics and Space Administration (NASA) defines software quality as "(1) the degree to which a system, component, or process meets specified requirements, and (2) the degree to which a system, component, or process meets customer or user needs or expectations [IEEE 610.12 IEEE Standard Glossary of Software Engineering Terminology]" (NASA, 2009, p. 1). According to NASA's definition of quality, two objectives should be the focus of a software development project: first, conformance to 'specified' requirements, which may be broader than customer requirements; and second, the artifacts and process should satisfy the customer or user needs.

ISO/IEC 25020 (2007) defines software quality as *internal quality* and *external quality*. *Internal software quality* is the capability of a set of static attributes of a software product to satisfy stated and implied needs when the software product is used under specified conditions; and *external software quality* is the capability of software products to enable the behavior of a system to satisfy stated and implied needs when the system is used under specified conditions.

In the internal software quality, the *static attributes* refer to the attributes that are related to the software architecture, structure and its components, which can be verified by review, inspection and/or automated tools (ISO/IEC 25020, 2007). In the external quality, the number of failures found during testing indicates external software quality measure related to the number of faults present in the program. The attributes of the behavior can be verified and/or validated by executing the software product during testing and operation (ISO/IEC 25020, 2007).

Many scholars have suggested definitions for software quality. Generally, the definitions refer to software quality as "*the extent of a software product to meet its customer requirements*" (i.e. Issac et al., 2006; Kitchenham and Pfleeger, 1996; Xenos and Christodoulakis, 1997). This thesis adopts the ISO/IEC 25020 (2007) definition.

Software quality construction is a complex socio-technical process (Hovenden et al., 1996), through which software quality is implemented into the software product throughout the software development life cycle (SDLC) (Busch et al., 2014). According to Hovenden et al. (1996), software quality is a result of many factors, including people, tools, methods, processes, management, techniques and quality assurance, working together to attain quality goals. According to Kitchenham and Pfleeger (1996), meeting quality goals refers to meeting customer requirements. In this regard software quality construction is a joint strategy that involves both technical and nontechnical stakeholders towards achieving predetermined quality goals.

In a study of software architecture descriptions, Smolander (2002) argues that system development projects are probably doomed to fail technically and organizationally if there is no proper communication and understanding among the stakeholders. The challenge in communication emanates from varying understanding and experience among the stakeholders. The same applies to software quality construction - it is difficult to define the required quality because it is sought by a diverse set of stakeholders with varying skills and experience, including technical stakeholders such as software engineers, systems analysts, and stakeholders at the customer side such as sales persons, end-users, and the public at large (Smolander, 2002). For a particular software product, the stakeholders must come into mutual understanding of the requirements and build the software to meet such requirements.

Park et al. (2012) relate organizational structures, communication and productivity in companies. According to Park et al. (ibid.), organizational structures contribute to operational efficiency within the functional teams. On the other hand, organizational structures can also lead to a lack of communication between the functional teams, making the company slow and inflexible. According to Hovenden et al. (1996), software quality construction can be regarded as an information-intensive process that requires involvement of the customer, management and other stakeholders in the development process for the purpose of enquiring, understanding, and building software artifacts that will address the specific needs of the customer or the company. The extent and type of stakeholder involvement depends on the type of the stakeholder, the type of the project and discretion of the developers (Lagrosen, 2005).

In the light of the literature above, I define the software quality construction process as deliberate actions taken by various stakeholders to define the objectives of software development, identify the necessary quality characteristics, and set strategies to create such quality into the software through processes such as requirement elicitation, design, implementation and testing.

2.3 Quality construction in software development

This section contains a brief description of software quality construction in the Software Development Life Cycle (SDLC), and the role of humans and software companies in the software quality construction process.

2.3.1 Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a series of distinctive activities which may take place sequentially or iteratively, depending on the methods adopted in the software development process (Busch et al., 2014, Tayntor, 2003; Trivedi, 2013). SDLC

consists of six basic steps: *planning, requirement elicitation, design, implementation, testing,* and *deployment*. Different sources use different terms for the steps.

Planning refers to preliminary activities undertaken before the actual software development process begins. In this stage various activities such as budgeting, scope of the project, objectives of the project and contract management are discussed. There is no generally accepted specific way of doing planning activities; project managers implement the planning as they wish. Wu and Simmons (2000) describe three general project-planning approaches: *past experience*, *standard guidelines*, and *support tools*. For example, Wu and Simmons (ibid.) argue that the use of a specialized tool, such as Software Project Planning Associate (SPPA), may be of good help especially for managing large software development projects, because it is hard to visualize software, and many unforeseen factors can affect the progress of the software project. Furthermore, the project managers choose whom to involve in the project planning. In some companies, managers involve technical teams (i.e. developers, testers, analysts, etc.) from the planning stage onwards while others involve technical teams at the later stages (*Publication II*). Basically, planning is one of the important duties of the project managers that may determine the success or failure of the project.

Requirements elicitation is a technical and specialized activity that aims at identifying, collecting, analyzing and prioritizing software requirements. Requirements elicitation is a critical process in quality construction and software development process in general (Dieste et al., 2008; Jung et al., 2014). Unless the actual requirements are elicited, it is difficult to develop a software product that will meet specific customer needs (Dieste et al. 2008). In this regard, requirements elicitors need to have a broad picture and experience of the domain where the software is needed. For example, Jung et al. (2014) argue that the requirements elicitation process is influenced by the culture and technical skills of the persons involved. Requirements elicitors may include development teams (designers, testers, programmers, etc.), marketing teams, R&D teams, or a special trained team.

It is difficult to discover all requirements at the beginning of software development projects (Dieste et al., 2008). However, Terzakis (2013) discusses the importance of requirements management from the beginning of the project. He emphasizes that there is a difference between randomly elicited requirements and those elicited systematically. Well-organized and documented requirements result in higher product quality, regardless of the method of development and increase of the complexity of the product (Terzakis, 2013). The discovery of requirements is a continuous process throughout the SDLC and beyond after the software has been delivered to the customer (Dieste et al., 2008). It is not necessary to have new requirements identified, as the old requirements may be redefined and refined. The process of redefining and refining requirements can be regarded as *requirements optimization*. Generally, it is not

possible to build all requirements elicited into the software product because of cost and time (Boehm, 1984). The requirements should be selected and prioritized. The process of requirements prioritization reduces the number of elicited requirements, but also increases the quality of the software product by identifying high value requirements. Figure 1 depicts the requirements prioritization and optimization process. Even though the requirements are meant to satisfy a specific customer, the goal should incorporate both business goals and software company goals (Savolainen et al., 2007). For example, the base of the triangle in Figure 1 represents the idea that a huge amount of requirements is elicited during requirements elicitation, but not all customer requirements will fit into business requirements, so some requirements may be dropped as a tradeoff between specific segments of customers or users. The double-headed arrows in the triangle show the agreement of customer and business requirements.

The second level in the triangle (Figure 1) entails the integration of customer and business requirements into general software requirements. After that, the software specifications are derived from the software requirements. In the final stage, the software product is developed based on the specifications obtained. The vertical double-headed arrow outside the triangle (Figure 1) indicates that quality construction is not a one-direction process through the SDLC. The discovery of requirements, improvement of design, coding, and testing process is continuous and iterative, throughout the development process and even after delivering the product to the customer.

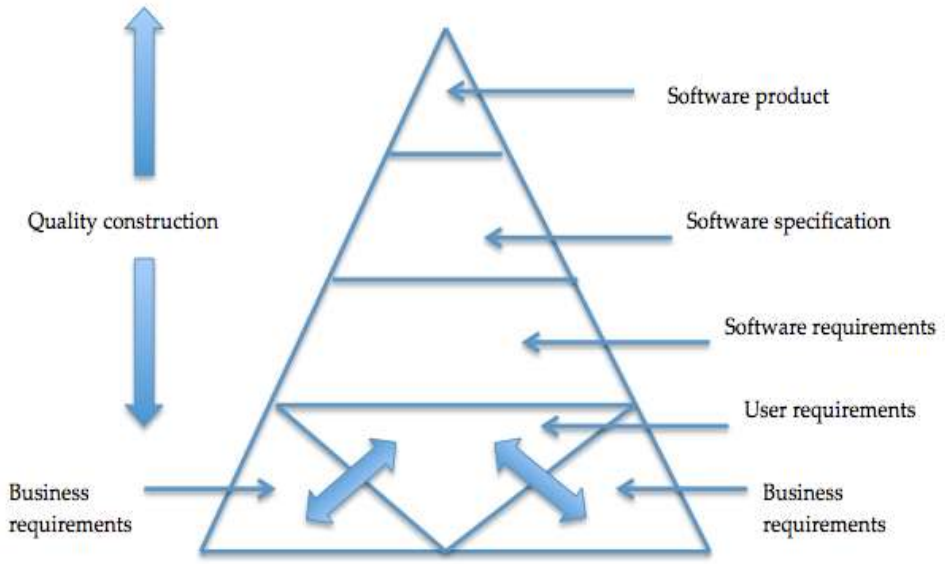

Figure 1. Optimization of requirements for a software product (*Publication I*)

The design, implementation, and testing processes of the SDLC do not depend on the method of software development e.g. agile or plan-driven (Salo and Abrahamsson, 2008; Terzakis, 2013). The methods affect, for example, the duration and frequency of the processes. Many software-developing companies have moved from traditional waterfall methods to agile methods (Salo and Abrahamsson, 2008), though the transition from the traditional waterfall to agile methods has not been smooth (Lester, 2013; Madabhavi, 2014; Salo and Abrahamsson, 2008). Lester (2013) discusses several challenges of adopting agile methods including the "fear of the unknown" and agile process management. However, the study by Salo and Abrahamsson, (2008) investigating the adoption of XP and scrum agile methods indicates that the methods are useful. In agile methods, the design, implementation, and testing processes take place iteratively (Tselentis, 2014). One of the advantages of agile methods is that the design can be refined along the implementation and testing processes. However, companies still choose to adopt waterfall methods in some of their projects, and carry out agile methods in others.

2.3.2 The role of humans and software companies in software quality construction

Software quality construction process is not the same as software development process (i.e. the activities of the SDLC). The software quality construction process involves more activities than the SDLC activities. One of those activities is to understand the humans so as to address their 'actual' needs (Christoffer and Furuknap, 2009). For example, Christoffer and Furuknap (ibid.) describe software development as an *art* through which a developer figures out what is in the mind of the customer and produces code, style, fashion, feature, pattern, design, etc., to meet the needs of that customer in the course of software development. Software is developed for users or customers, who are mostly not technical, so some of their requirements are represented in a fuzzy manner. However, there should be a common understanding and cooperation between the developers and customers throughout the development process for the purpose of understanding the requirements, and developing right software to address the specific customer needs (Dieste et al., 2008). Thus, software quality construction involves activities within and outside companies, including skills beyond the computer science discipline working together towards achieving the predefined quality goals.

Software quality construction process can be influenced by several factors (Hovenden et al., 1996). The scope of this thesis is confined to the factors surrounding humans (i.e. developers and customers) and software companies. Therefore, this study is focused on the activities conducted within software companies. The information regarding customers has been studied through the interaction of customers and software companies in the course of software development process. In the scope of this thesis, the factors that influence software quality include *customers* (i.e. individuals and businesses), *developers* (i.e. testers, programmers, designers, architects, etc.), *organizational structures* of software companies, *tools*, and *processes*. According to Carnegie Mellon University, Software Engineering Institute, CMU/SEI-2010-TR-033 (2010), there are three critical dimensions that companies focus on during quality construction: people, procedures and methods, and tools and equipment. However, the customer is a fundamental entity in software quality construction because quality is judged and defined from the customer's perspective (ISO/IEC 25020, 2007). Software that does not address the particular needs of the customer is not quality software, even though it may be well built from the point of view of the developers. In this thesis, the term customer refers to individuals or businesses that use software products for various purposes.

Software companies vary in size, resources and specialization. However, all companies have common features, such as the presence of management and organizational structures, which allow easy comparison between the companies. The

aim of this study is not to compare companies but to investigate software development activities within the companies to understand how software quality is constructed. For example, each software project has developers and project managers. The managers in software companies manage resources (humans, tools, fiscal, etc.) and processes (CMU/SEI-2010-TR-033, 2010). Managers decide on the mode of operation and the organizational structures of their company, which implies that management can influence the development process and the quality of products. According to Park et al. (2012), organizational structures have an impact on the productivity of the functional teams within the company, and hence affect the quality of the products. Although the findings presented by Park et al. (2012) are not specific for software companies, the study presented in this thesis indicates that software companies are also vulnerable depending on the type of organizational structures (*Publications III* and *IV*).

The Capability Maturity Model (CMM) (CMMI, 2012; Paulk, 1995; Paulk et al., 1993) suggests that the quality of a product is directly related to the quality of the engineering process that the company has reached. However, Kitchenham and Plefeeger (1996) argue that there is little evidence that conformance to process standards guarantee good quality. According to Suominen and Mäkinen (2013), software is intended to solve problems in a complex socio-technical setting. So, the success of the project depends on the management of the processes and software design that is domain-specific at the business level, than relying on processes and methods alone. Thus, managers in software companies should have skills and experience of their customers' business environment and the domain to be able to make sound decisions on resources and processes, so as to achieve the desired quality goals (Boehm and Ross, 1989; Colomo-Palacios et al., 2013).

2.4 Human factors in software quality construction

The literature suggests that software quality can be improved through improving the capability of organizational processes (CMU/SEI-2010-TR-033, 2010) and improving the capability in the individual development processes (ISO/IEC 15504-5, 2012). Furthermore, the Software Engineering Institute (SEI) suggests that companies should focus on people, methods and tools as critical factors in improving business. According to Petre (2010), the success of projects depends on how well the humans can reason and perform tasks despite of the growth of technology (tools, methods, etc.). In this regard, humans play a primary and fundamental role in quality construction.

The works of Boden (2004) and Shneiderman (2007) suggest that software development is a mental activity that starts with an idea (*creativity*) from the

developers or customers. In the real business environment, it is a challenge to meet both customer requirements at a competitive price yet meet the time and desired quality; *innovation* is one of the success factors providing competitive edge (D'Aniello et al. 2006, Strecker 2009, Edison et al. 2013). According to Miatidis et al. (2008) and Shneiderman (2007), the *experience* of developers in a particular domain and the time spent in solving various problems associated with customers, methods, requirements, etc. in that domain, has a positive impact on quality construction. Technical skills, such as programming, software testing, etc., are mandatory but the experience in the domain enables the developers to improvise suitable solutions because of their domain knowledge.

Customer satisfaction has never been an easy issue. It is not enough to develop a working product but a product that is also attractive to the customer. This means presenting a product in a manner that the customer will appreciate it. Christoffer and Furuknap (2009) and Patre (2010) discuss the importance of *art*. The art of design, artifact creation, etc. have a positive influence on product quality.

This study finds creativity, innovation, experience and art as important human factors that influence software quality construction. Therefore, this section presents an insight into creativity, innovation, experience and art in software quality construction.

2.4.1 Creativity

Creativity is "the ability to come up with ideas or artifacts that are new, surprising and valuable" (Boden, 2004, p.1). Piffer (2012) argues that a product that is novel, but not usable cannot be considered creative. This view expands the definition of creativity to include the item "usefulness" of the new idea. On the other hand, Arden et al. (2010) and Zeng et al. (2011) expand the definition of creativity further to include the concept of "beauty" (arts) and "appropriateness". For example, Zeng et al. (2011, p.25) define creativity as "the goal-oriented individual/team cognitive process that results in a product/service that, being judged as novel and appropriate, evokes people's intention to purchase, adopt, use, and appreciate it". Thus, on the basis of the literature, I define creativity as the *use of original ideas to create a useful, appropriate and attractive (artistic) product that will evoke people's intention to purchase, adopt, use, and appreciate it*. However, Piffer (2012) argues that it is not only difficult to have one definition of creativity but it is also difficult to measure it by one scale. He further declares that creativity is the sum of accomplishments of some tasks or products or services, which can distinguish one product from others. According to several definitions of creativity (Arden et al., 2010; Boden, 2004; Piffer, 2012; Zeng et al., 2011), software quality can result from developers' creativity and satisfy the customer even though he was not involved in the requirements elicitation.

2.4.2 Innovation

According to Strecker (2009), *innovation* is the implementation of new factor combinations (e.g. new goods, new production methods) that lead to significant improvements of existing methods.

The goal of software quality construction is to develop a software product, or improve an existing product for the purpose of satisfying customers (Strecker, 2009). According to Edison et al. (2013), innovation can be categorized into four areas: product innovation, process innovation, market innovation and organization innovation. In order to develop a quality product or improve an existing product, innovation is important. For example, Edison et al. (2013) define *product innovation* as the creation and introduction of new (technologically new or significantly improved) products that are different from existing products as regards architecture, structure, technology, features or performance.

The improvement of products may start by improving the process used in the development process. Edison et al. (2013) define *process innovation* as the implementation of a new design, analysis or development method that changes the way in which products are created. Recognizing the importance of other roles in addition to technical teams, Edison et al. suggest that there should be innovations on marketing strategies. Edison et al. describe *market innovation* as the implementation of new or significantly modified marketing methods, strategies and concepts in the product design or packaging, placement, promotion, or pricing. It includes opening up new market opportunities, position innovations (including changes in the context in which the products are introduced) and the implementation of new or significantly modified marketing strategies. Finally, Edison et al. (ibid.) suggest *organization innovation*, which refers to the implementation of new organizational methods in the firm's business practices, workplace organization or external relations. It includes changes in the architecture of production and accounts for innovations in management structure, corporate governance, financial systems or employee remuneration systems.

Innovation refers to methods (Strecker, 2009) that lead to better or improved products. The customers' demand may be beyond the requirement elicited, because customers may represent their needs in arbitrary expressions or in a vague language that is difficult to interpret precisely. Some customers do not know what they need until they see it. Thus, the software quality construction process may tap into the advantages of innovation along with the involvement of customers for the purpose of meeting their needs.

2.4.3 Experience

Miatidis et al. (2008) describe *experience* as the knowledge background that concentrates mental stimuli from previous experience, education, know-how and guidelines stemming from the company, or even pure instinct, and enables a person to come up with optimal solutions in problem solving.

The Oxford dictionary (2005) defines *experience* as the knowledge or skill acquired over a period of time, especially that gained in a particular profession by someone at work.

Fagerholm and Münch (2012) suggest several approaches to addressing 'experience'. They consider 'experience' to refer to both immediately perceived events as well as the memories of events and the knowledge gained by interpreting and reflecting on remembered events.

The three views of experience in Fagerholm and Münch (2012), Miatidis et al. (2008) and Oxford dictionary (2005) suggest that experience adds value to a product. For example, developing software in the domain where developers have experience enables the developers to optimize solutions, and satisfy the customers better, because they have experience with their customers in that domain. Thus, it is important for the developers to have domain knowledge in addition to software development skills.

2.4.4 Art

Christoffer and Furuknap (2009) describe *art* as the personal ability of a developer to figure out what is in the mind of a customer and produce code, style, fashion, feature, pattern, design, etc. to meet the needs of that customer in the course of software development.

The Oxford dictionary (2005) defines *art* as the expression or application of human creative skill and imagination, typically in a visual form such as painting or sculpture, producing works to be appreciated primarily for their beauty or emotional power.

Schaefer (2009) considers design as art. He argues that art exists in the context of design and that it is manifested in the translation of ideas into tangible or visible phenomena of the artifacts. Art can be applied to create a new product or to make an existing product to look new, i.e., "*to make the familiar strange and the strange familiar*" (ibid., p. 26). Emphasizing art in relation to design, Schaefer elaborates that design includes the arts of graphic design, architecture, and product design. The art in the design is what creates the differences between designers and products, which hence dictate the quality of such products.

According to Christoffer and Furuknap (2009), art is a way of representation or expression of customers' specification in software in more meaningful and appealing manner (Oxford, 2005; Schaefer, 2009). Thus, art is an important aspect of software quality construction.

2.5 Software quality construction in the Buy Your Own Device (BYOD) trend

The Bring Your Own Device (BYOD), which is also termed as consumerization of IT, (Sangroha and Gupta, 2014; Scarfo, 2012), is a phenomenon where end-users bring their personal devices to their working places.

According to Scarfo (2012), the BYOD trend has gained increasing popularity in the past decade. This trend has also shaped the software and hardware industry whereby software and hardware companies try to reorient product and service designs around the individual end-users. Software development focuses on the individual consumer as the primary driver of product and service design.

The BYOD trend has led to the "ubiquitous computing" trend, which is characterized by the increased use of mobile devices such as smart phones, tablets, and other handheld devices, including wearable devices in working environments, learning environments and other places (Mahalingam and Rajan, 2013). In the BYOD environment some companies and businesses are forced to deliver services into mobile device platforms, which increasingly add to the challenge in software quality construction in terms of security issues and meeting quality requirements in general.

The BYOD trend has imposed new challenges to software industry. Software developers should think about both the hardware i.e. the devices and the business environment where the end-user of the device is working. For example, one of the challenges is to develop quality software at reasonable cost and time (Osterweil, 1997), yet to meet quality requirements, such as security for achieving organizational goals in the company where the end-user works (Scarfo, 2012). This means that although the software aims to meet end-user requirements, it should also meet some business requirements, which in this respect vary between the users, depending on the type of business they are engaged in. It is challenging to have a common goal between software developing companies, individual end-users of software products, device manufacturing companies and the companies where the end-users work. However, Barney and Wohlin (2009) argue that software quality construction should focus on achieving some common goals. Perner (2008) suggests that software-developing companies should optimize quality on the basis of their customers and add value that will also benefit the companies where the software or devices will be used in other

environments beyond personal use. On the other hand, Savolainen et al. (2007) discuss the importance of varying requirements. Savolainen et al. argue that the requirements should be adaptable in various situations for the purpose of providing key competitive advantage that allows economic success for the product line.

2.6 Summary

It is difficult to define quality, and there is no one ultimate definition of quality agreed upon by all stakeholders (Garvin, 1984; Seth et al., 2012; Kitchenham and Pfleeger, 1996), and nor is there any specific approach to achieving software quality (Kitchenham and Pfleeger, 1996), or a universal measurement or scale for quality (Jørgensen, 1999). For example, Kitchenham and Pfleeger (1996) describe quality construction as aiming at an illusive target. Although tools and processes are crucial, humans, companies' management teams, and organizational systems (i.e. organizational structures, management of resources, relationships among the functional teams, etc.) have a positive impact on the quality of the products. The quality of a software product cannot be implemented in the development (coding) process only; it is the result of the joint efforts of all functional teams within the software companies, and careful involvement of customers in the necessary stages of development throughout the SDLC. In certain situations software developers adapt to the market trends and not standards because the pace of technology is faster than the change of standards (Kalyani, 2013). Customer requirements are often more flexible and varying than standards; therefore innovation is required to meet such requirements (Aslhford, 1985). In the BYOD trend, the development of software products and services is reoriented to individual customers, which imposes the challenge of meeting requirements for both the software customer and the place where the customer works. Thus, software companies need to use varying requirements that are adaptable into various situations (Savolainen et al., 2007).

3 Research problem and methodology

The author had a privilege to work in the research project called 'Software Testing for Intended Quality (STX)'. This was a three-year project started in August 2011. The research goal for STX was to explain how software development, testing and quality depend on one another. From this objective and studying the literature, the research problem of this thesis was identified and pursued. During the study, which was conducted in software companies on the software development and testing process, 'software quality construction' was suggested as the research topic among several other topics that were studied in parallel.

In this chapter, the research problem and its shaping, the research methods, and the research process are described. The chapter also describes the motivation and the rationale of employing the selected research approaches of the study.

3.1 The research problem

Software engineering has evolved and faced many developments since the 1940's. Numerous researches and studies have been conducted in the area of software engineering, delivering many useful suggestions for improving the quality of software products and services. Despite the many initiatives and developments such as quality standards, quality models, best practices, methods, and advancement of technology in the software industry, software quality is still a challenge in the 21st century. The advancement of methods and tools for developing and testing software has not been able to eradicate software quality issues. Software testing is yet reported to cost up to 50% of the project cost (Kit 1995).

Companies are investing billions of dollars each year in research and development (R&D) in order to improve products. For example, Hartung (2012) presents a list of top 20 spenders on R&D in the year 2011. The mentioned companies include IT companies such as Microsoft (9.0$ US billion), Samsung (9.0$ US billion), Nokia (7.8$ US billion), Intel (8.4$ US billion), Panasonic (6.6$ US billion), IBM (6.3$ US billion), and Cisco (5.8$ US billion). However, Hartung (2012) argues that despite the heavy investment in R&D, companies face challenges continuously.

Criticizing the expenditure of huge sums of money on historical programs, following historical patterns and trying to defend and extend the historical business, Hartung (2012) argues that innovation is the right path. Instead of looking deeper in the products, companies need to look wider. They need to investigate alternative solutions, rather than more of the same. This view suggests that customer requirements are changing, and the products should change likewise (Hartung, 2012). Cockburn and Highsmith (2001) argue for the human factor in the software development process. They emphasize individual competence in the development teams as a critical factor in software product development and software product success. For example, different designers with different 'art of design' will produce different software products in terms of quality, despite of using the same equipment and resources in the same environment. Investment in innovation and in development teams will mark the difference in the quality of products (Cockburn and Highsmith 2001).

The goal of the present research is to investigate the role of humans and software developing companies in the software quality construction process. The study seeks to understand (1) the human factors inside and outside software companies that influence software quality; and (2) organizational factors that influence software quality. In particular, the research focuses on activities in software developing companies. Stakeholders outside software companies such as customers are studied through their interactions with the companies.

The research problem has been decomposed into sub-problems, which are addressed in more detail in *Publications I-V*. Table 1 shows the mapping between the research problem, the objective of the study and the publications.

Table 1. Decomposition of the research problem

Sub-problem	**Objective of the study**	**Publication**
1. What is the role of requirements, stakeholders, and resources in software quality construction?	Understanding the role of requirements, stakeholders and resources in software quality construction.	*Publication I*
2. What is the role of human factors in software quality construction?	Studying the role of human factors inside and outside software companies in software quality construction.	*Publication II*
3. (i). What are the organizational challenges that affect software testing? (ii). What are the challenges related to customers that affect software testing?	Understanding the organizational and customer-related challenges that affect the software testing process in software developing companies and how these challenges affect software quality construction.	*Publication III*
4. What are the management issues that influence efforts towards software quality construction?	Understanding the management issues that influence the efforts towards software quality construction.	*Publication IV*
5. How do software product customers influence software quality construction in a BYOD environment?	Analyzing the challenges of meeting the quality requirements of software products in the BYOD environment.	*Publication V*

3.2 Research methodology

Software engineering is a discipline that cuts across many other disciplines, including science, art, psychology, and engineering (Ambler, 2009; Cross, 2001; Marcos and Zagalo, 2011; Schaefer, 2009). Software stakeholders come from different backgrounds,

experience and knowledge (Hirschheim, 1985; Hovenden et al., 1996; Smolander, 2002). Studying their needs and developing software to meet their needs requires skills and knowledge that are beyond the computer science body of knowledge (Boehm and Ross, 1989; Colomo-Palacios et al., 2013). For example, knowledge of the business process or environment and the domain in which the software is applied has a positive impact on quality construction. In this regard, a variety of research methods can be employed to research problems in the field of software engineering (Shaw, 2002). However, the researcher should choose an appropriate method to acquire better results from the study (Easterbrook et al., 2008).

Several research methods exist, which can be classified into two major categories: *quantitative* and *qualitative* methods. The quantitative methods are based on the quantities or the frequencies of occurrences of an event, to signify the relevance of an observation or a finding (Wohlin, 2006). The advantage is that quantitative data promotes comparisons and statistical analysis (Wohlin, 2006). On the other hand, qualitative methods determine the relevance of an event or a finding by its gravity, and the quality of the finding grounded on the data (Van Manen, 1990; Klein and Myers, 1999).

Järvinen (2004) classifies research methods into ten approaches, and names the classification the taxonomy of the research approach (Figure 2). In the taxonomy, the top-down principle is applied, which means that each research approach is first divided into two classes, and one or both are then divided again into two sub-classes, etc.

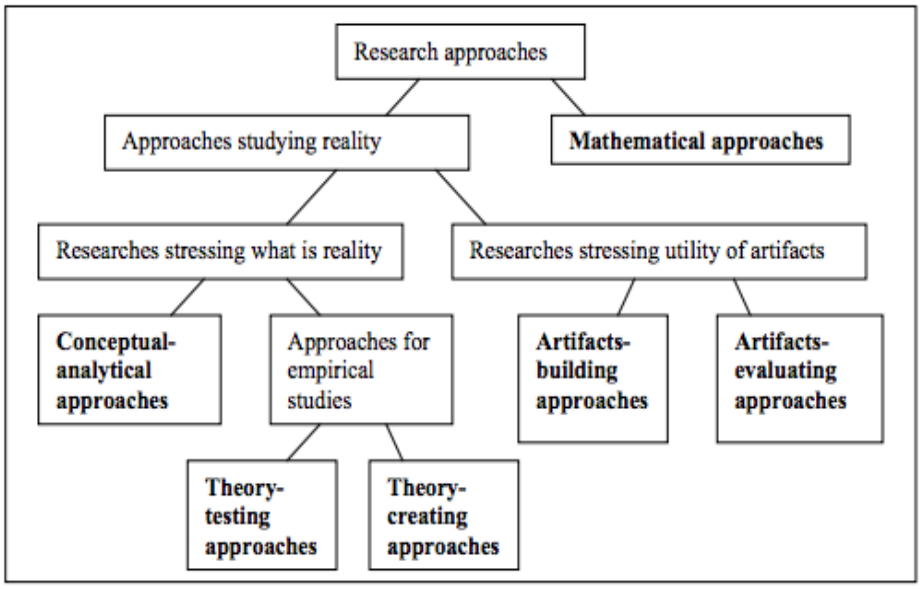

Figure 2. Järvinen's (2004) taxonomy of research methods

At the beginning of the classification, the mathematical approach is differentiated from studies of reality, because mathematics uses symbols and formal language, algebraic representations, etc. without having direct reference to objects in reality. The rest of the approaches are concerned with studying reality. The approach for reality is divided into researches stressing what is reality (i.e. conceptual analytical approaches or approaches for empirical studies) and those which try to find out the utility of artifacts (i.e. artifact-building approaches or artifact-evaluating approaches). The approaches for empirical studies are further divided into theory testing and theory creating approaches. In the software engineering field, there are very few established and comprehensive theories that can be tested with empirical observations (Jørgensen, 1999). Therefore, this thesis adopts qualitative empirical research methods with the aim of theory creation.

3.2.1 Theory creation

According to Easterbrook et al. (2008), the confusion around the selection of empirical methods and appropriate evaluation of empirical research occurs because the underlying philosophies are never mentioned. It is difficult to conduct research and judge its results without some criteria for judging the validity of the method and the results thereafter. So, understanding the philosophical viewpoints of the selected research methodology is mandatory. There are four popular philosophical viewpoints in knowledge and theory creation: *positivism*, *constructivism*, *critical theory* and *pragmatism* (Easterbrook et al., ibid.).

- **Positivism**: Positivists believe that knowledge is acquired through the process of verifying theories by testing hypotheses derived from them, and that is why studies are designed to test theories through hypotheses. For example, the survey and some case study research methods are based on the positivist philosophical viewpoint. (Easterbrook et al., 2008)
- **Constructivism**: Constructivists believe that knowledge is socially created and it emerges from the data, and only the data explains the validity of the emerged theory (Berger and Luckmann, 1967; Easterbrook et al., 2008). Constructivism is also known as interpretivism (Klein and Myers 1999). Easterbrook et al. (2008) argue that constructivists concentrate on understanding how different people make sense of the real world, and how they assign meaning to actions, and therefore theories may emerge from this process. However, the theories emerged cannot be separated from the context being studied. Methods closely associated with constructivism are e.g. ethnographies, exploratory case studies and survey research (Easterbrook et al., ibid.).
- **Critical theory**: According to Easterbrook et al. (2008, p. 6), critical theorists believe that "theories are assertions of knowledge (and therefore power), to be critiqued in terms of how they shape that power". Critical theorists take the role of

emancipation and advocacy by challenging the existing perceptions about common practices to draw attention to things that need changing (Easterbrook et al., ibid.). Critical theory is built on three elements (Myers and Klein, 2011): *insight*, *critique* and *transformation*. The first element, insight, aims at providing an insight into the background, and provides understanding of the phenomenon under investigation. The second element, critique, provides a critical viewpoint to the phenomenon in its present context; and the third element, transformation, is improvement of earlier known constructs, ideas and existing social theories (Myers and Klein, ibid.). Methods closely related to critical theories are case studies and action research (Easterbrook et al. 2008).

- **Pragmatism**: Pragmatists believe that "theories are the products of a consensual process among a community of researchers, to be judged for their practical utility" (Easterbrook et al., 2008, p. 6); and therefore knowledge is judged by how useful it is for solving real-world problems (Easterbrook et al., 2008). Pragmatism advocates relativism in the sense that truth is relative, so consensus is the key in avoiding obvious criticisms. In the pursuit of knowledge, pragmatism values practical knowledge over abstract knowledge, and uses whatever methods are appropriate to obtain it. Thus, pragmatists strongly prefer mixed methods research, where several methods are applied to investigate the phenomenon under study (Easterbrook et al., ibid.).

This study follows the constructivist stance. The grounded theory (Strauss and Corbin 1990) employed allows systematic data collection and analysis. During data analysis, similar concepts are grouped in the coding process, and the phenomena are studied based on their relationships established in the categorized data. Hence, the researcher studies the relationships to uncover the undelaying meaning and draws inference from the established relationships, which is stated as a hypothesis or a theory that may be applicable in the studied context (Easterbrook et al., 2008).

3.2.2 Research subject

The ISO/IEC 25010 (2010) (System and software quality models) and ISO/IEC 25020 (2007) (Measurement reference model and guide) standards were used initially to understand and approach the research subject of this thesis. ISO/IEC 25020 (2007) provides software quality definitions, software quality characteristics, and a guide to software product quality requirements and evaluation. ISO/IEC 25010 (2010) describes software quality from different stakeholder perspectives, and also provides quality models such as quality model framework, quality model, quality-in-use model, product quality model, and explains the relationship between the models.

ISO/IEC 25010 (2010) defines software stakeholders, and categorizes them into three major groups: the *primary user*, the person who interacts with the system to achieve

the primary goals, *secondary users* who provide support; and the *indirect user*, the person who receives output, but does not interact with the system.

The five stages of the Software Development Life Cycle (SDLC) (Busch et al., 2014, Tayntor, 2003), i.e. planning, requirements elicitation, design, implementation and testing, were used to study the activities involved in the software development process.

3.2.3 Selection of the research method

Quantitative vs qualitative methods: Empirical studies can be divided into two major categories: quantitative and qualitative studies. Quantitative research methods collect numerical data and analyze it using statistical methods. Qualitative methods collect qualitative data such as text, images, video, and sound through observation, for example interviews, and analyze it using qualitative data analysis methods (Moody, 2002). According to Moody, quantitative methods are more appropriate when theory is well developed, and for the purposes of theory testing and refinement; when the subject area is not well understood, qualitative methods are more appropriate. As this study explores the quality construction process, where no clear theories and hypotheses exist (Jørgensen, 1999), qualitative research methods are more suitable to conduct this study.

Case studies and the grounded theory: According to Järvinen taxonomy (2004), this thesis is part of studying reality. The goal is to conduct an empirical study in software developing companies to investigate software development activities in their natural contexts (Eisenhardt, 1989). To achieve this goal, the case study and grounded theory empirical methods were chosen because both methods can be used for theory creation (Yin, 2002; Glaser and Strauss, 1967).

Why is the grounded theory an appropriate method for this research?

i. Grounded theory aims at creating a theory from the data without using a priori concepts (Glaser and Strauss, 1967; Mayer, 2001; Strauss and Corbin, 1990), and so it is suitable for uncovering the unknown reality.
ii. Grounded theory provides a systematic approach for theory creation (Strauss and Corbin, 1990).
iii. As software quality construction is a social-technical process (Hovenden et al., 1996), the grounded theory provides a comprehensive description of the social process being studied (Jones et al., 2005).
iv. Grounded theory has the capacity to interpret complex phenomena (Charmaz, 2010).
v. Several studies suggest the grounded theory as an appropriate method for conducting qualitative studies (Arshad et al., 2013; Jones and Alony, 2011;

Mayer, 2001; Moody, 2002; Smolander, 2002; Taipale and Smolander, 2006).

The grounded theory has been used successfully in diverse areas, including information systems (Bryant, 2002), software engineering (Smolander, 2002), healthcare (Akbar, 2003), film industry (Jones, 2005), and modeling (Dal Forno and Merlone, 2012).

3.3 The grounded theory

Initially, Glaser and Strauss (1967) established the grounded theory research method in 1967. The grounded theory is an inductive, theory-building methodology that allows a researcher to develop a theoretical generalization on the observed phenomenon by grounding the emerging theory on the empirical data (Martin and Turner, 1986). The grounded theory provides a systematic method of analysis, which allows the researcher to visualize the emerging concepts, and their relationships, through which the researcher explores the research area, and allows issues to emerge. Consequently, the grounded theory is useful for providing insight into areas that are relatively unknown by the researcher (Strauss and Corbin, 1990; Jones and Alony, 2011). The background of the grounded theory is in social sciences, and the approach allows researchers to interact with the subject under study by combining various data sources without restricting data formats (Joan and Pastor, 2007). The grounded theory is suitable for the investigation of phenomena in their natural context and allows the researcher to build a theory from data analysis.

After publishing the grounded theory in 1967, Glaser and Strauss differed in their opinions on the role of emergence in theory creation. This conflict gave birth to fundamental schools of grounded theory, "Glaserian" grounded theory (Glaser, 1992) and "Straussian" grounded theory (Strauss and Corbin, 1990). There are many, although minor, differences between the two schools of grounded theory. For example, Glaser emphasizes the emergence of theory, when the researcher has an empty mind; Strauss permits a general idea of the area of the study, and a systematic approach in theory creation. In this view, Glaser criticizes Strauss for forcing the theory instead of letting the theory to emerge (Jones and Alony, 2011). However, Strauss and Corbin (1998), and Corbin and Strauss (2008) agree on the broad purpose of the grounded theory. They acknowledge that all studies will not produce a theory and admit that the grounded theory may also be used in some studies to produce useful descriptions. Thus, Strauss and Corbin acknowledge that grounded theory techniques have uses beyond theory building.

This thesis follows the "Straussian" grounded theory (Strauss and Corbin, 1990), because it provides a structured approach to the analysis process and is therefore

suitable for team research efforts. Another reason is that it allows the researcher's personal and professional experience together with the read literature to affect his theoretical sensitivity. Theoretical sensitivity grows during the study and helps in the formation of a hypothesis or a theory (Joan and Pastor, 2007).

The approach in theory creation is built on two main concepts: constant *comparison of concepts* and *theoretical sampling* (Eisenhardt, 1989). Every piece of new information and concept is constantly compared to the collected data to uncover similarities and differences. The process of data collection and analysis take place concurrently with every identified new lead in the data. The new leads are included in the next data collection round, until a saturation point is reached. Theoretical saturation is the state whereby the additional data collected does not provide any new knowledge about the studied phenomena (Glaser and Strauss, 1967; Strauss and Corbin, 1990). The disadvantage of theories built through the grounded theory is that they are dynamic and not static. With the increase of data, the theory may change (Strauss and Corbin, 1990).

In the grounded theory, coding is the fundamental process for data analysis and building a theory. The Straussian grounded theory emphasizes systematic codification and categorization of concepts in the data. The process of data analysis involves three stages of coding: *open*, *axial* and *selective* coding. Open coding is an analytical and interpretative process, where data is broken down into distinctive concepts and labeled (Strauss and Corbin, 1990). The aim of open coding is to identify the concepts in the data and group them according to their similarities and differences. The process of grouping data is called categorizing.

In axial coding, the relationships between the categories are identified and studied. The aim of axial coding is to understand the underlying meaning of the categories so that dependencies and relationships can be set between the categories (Strauss and Corbin, 1990). Open and axial coding can be done parallel because the developed relationships between the concepts are likely to lead to new concepts and relationships (Strauss and Corbin, ibid.). Each phenomenon represented by a category is given a conceptual name. The open and axial coding processes continue until no new concept or category emerges from the data with added new information. This state is referred to as theoretical saturation.

Selective coding is the process of integrating the categories to obtain the core category. The core category could be one of the identified categories, if it is board enough to describe all other categories. If there is no such broad category, then a conceptual category is created and given a name that will describe the rest of the categories. The purpose of establishing a core category is to narrow down the concepts into a hypothesis or a theory (Strauss and Corbin, 1990).

To summarize, this thesis includes three phases of the research project, all of which employ the grounded theory intensively. The firsts phase can be considered as a case study because of a few a prior concepts, such as the ISO/IEC view of software quality, which gave us the initial direction to engage in the study; then we collected the data from practitioners, and used the grounded theory analysis methods that led us to discover new concepts and hypotheses. The second and the third phases followed the grounded theory principles. In the beginning it seemed difficult for me to use the grounded theory. I was comfortable to proceed with the methodology because I was working with a research team whose members are highly experienced with the grounded theory.

3.4 Research process

In this section, the three phases of the research process are presented in detail. The data collection and analysis processes, together with finishing and reporting of the study are also explained.

3.4.1 Research phases

The beginning of the research included studying important constructs, such as software quality definitions and quality characteristics (ISO/IEC 25020, 2007), understanding earlier research on the subject, selection of cases, crafting of instruments and setting strategies for data collection (Eisenhardt, 1989; Strauss and Corbin, 1990). Although the theory creation through the original grounded theory (Glaser and Strauss, 1967) requires no a prior concepts or hypotheses, Eisenhardt (1989) and Strauss and Corbin (1990) recognize the importance and contribution of a prior constructs in sharpening the emerging theory.

The research process was divided into three phases (Figure 3). In Phase 1, we conducted one study that produced *Publication I*. The study explored how practitioners perceive, plan and implement quality in software products. The study identified the role of requirements, stakeholders and potential (i.e. human factors) of the developers in the software quality process. The results obtained in this study provided an insight into the factors, other than tools and methods, which need attention in the software quality construction process.

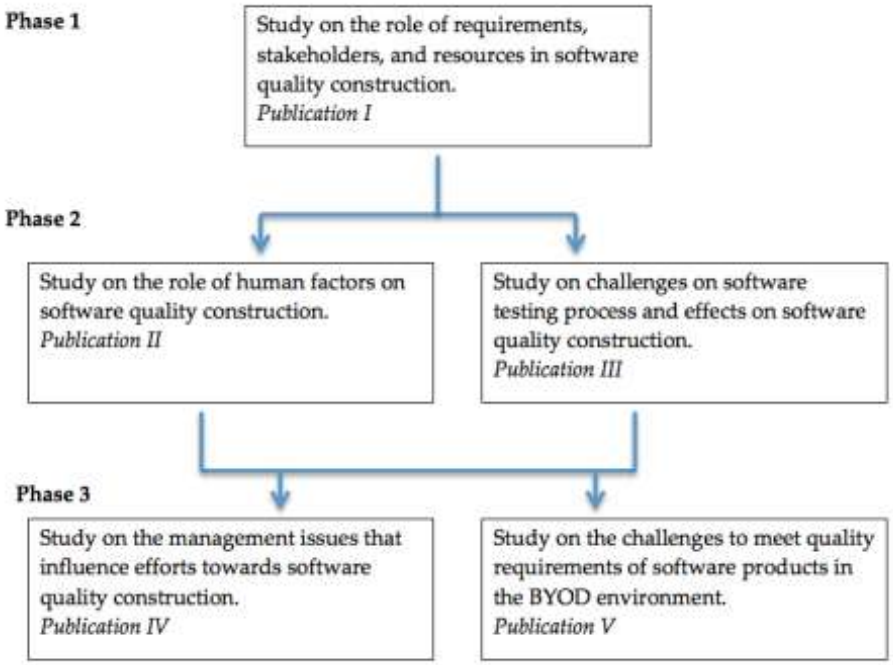

Figure 3. The three phases of the research process

In Phase 2, we considered the leads in the data which were obtained during Phase 1 and modified the research questions so as to gather more relevant data to pursue the research objectives (Strauss and Corbin, 1990). We conducted a study focusing on human factors; the results produced *Publication II*. Furthermore, we conducted an investigation on software development activities, focusing on software testing, and studied the management issues that have direct or indirect effect on software quality. This study generated *Publication III*, which extends the results obtained during phase 1.

During Phase 3, we pursued further the new data leads identified during Phase 2. Few phenomena indicated that organizational management, such as project managers, etc., and the organizational structures have an influence on software quality. So, the study in Phase 3 produced *Publication IV*, which addresses organizational factors, including management issues that influence the software quality construction process. Furthermore, observations in the data indicated that the BYOD trend brings challenges to software quality construction, so the study also produced *Publication V*.

3.4.2 Data collection

During the research project, most of the data was collected with semi-structured interviews. The interview questions are available at http://www2.it.lut.fi/project/STX/material.html. In some companies, there was additional information collected during interviews, for example presentations and written materials such as pamphlets and brochures. In some instances, a lengthy discussion started after the interview sessions, adding valuable information on the research subject. A total of 40 interviews were conducted in 13 software companies. Phase 1 involved 18 interviews, Phase 2 involved 15 interviews and Phase 3 involved 7 interviews. The length of an interview varied from 55 to 65 minutes. All the interviews were tape-recorded and transcribed. In total the transcription consisted of 480 A4 pages, in Times New Roman font 12.

The software companies involved in the interviews included 6 small companies, 1 medium-sized company and 6 large companies. The company size classification is based on the SME European Union definition (EU, 2003), in which a small company is characterized as having fewer than 50 employees, a medium-sized company has fewer than 250 employees, and a large company has more than 250 employees. Both nationally and internationally operating companies were included.

Theoretical sampling was applied in identifying the case organizations (Pare and Elam, 1997). The goal of theoretical sampling is not to get a representative sample of all possible variations, but to gain deeper understanding of the analyzed cases and to identify concepts and their relationships (Eisenhardt, 1989; Taipale, 2007). We selected 13 cases out of the 26 contacted companies, following a polar-type strategy (Eisenhardt, 1989) to cover different domains, methods of development, type of software, and size of company. The polar-type strategy was selected because it can help researchers to explore, understand, and explain the variety of characteristics of the phenomenon.

The data analysis was conducted parallel to the data collection process. Every time a new lead was identified, the interview themes and questions changed slightly along the research process to follow the leads. The accommodation of new leads into the research process is allowed and it enhances the emerging theory (Eisenhardt, 1989); the grounded theory describes it as theoretical sensitivity (Strauss and Corbin, 1990). Table 2 presents the research phases and themes.

The interviewees were practitioners in software developing companies. However, a few interviewees were consultants who worked with multiple companies, and whose experience in diverse companies and of various domains increased the value of the information given. Phase 1 of the study included more companies than Phase 2, and Phase 3 included the least because we did not identify new concepts in some

companies. Companies with no new leads to follow were excluded from the next data collection round. This is also the reason why some companies had 4 interviews while some had only 1 interview. Table 3 presents the case companies, business domains, phases of the study (i.e. interview rounds), company sizes, and the roles of the interviewees.

Table 2. Phases and themes of the study

Phase	**Themes**
1	Software development, testing, quality and change management.
2	Software development, testing and quality management.
3	Human factors, organization and software project management.

Table 3. Business domains, interview rounds, company sizes and roles of interviewees

CASE	Business domain	1st Round interviews	2nd Round interviews	3rd Round interviews	Company size	Roles of the interviewees
A	Inventory management systems.	1			Small	R&D and quality assurance manager (1).
B	Banking and insurance.	4	5		Large	Test analysts (1), test designer (2), Designer and developer (2).
C	Space satellite.	1	1	1	Small	Designer and developer (2) and Project manager (1).
D	Web applications.	1			Small	Tester and developer (1).
E	Embedded software.	4	4	2	Large	Tester (1), developer (2) and requirement management (1).
F	Quality and testing consultancy	1		2	Medium	Quality manager (1), developer (1) and consulting tester (1).
G	Various software developers.	1	1		Large	Quality manager (1) and tester (1).
H	Cloud computing Web applications.	1			Small	CEO, developer, tester and designer (1).
I	Fleet management systems.	2	1		Large	Test consultant (1) and test manager (1).
J	Cloud computing services and consultancy.	1			Small	CEO (1).
K	Banking, energy, health, etc.	1	1	1	Large	Quality assurance and tester (1).
L	Development and testing consultants.		2		Small	Consultant tester (1) and developer (1).
M	Various software developers.			1	Large	Project manager (1).
13		**18**	**15**	**7**		

NB: *Case identifiers A-M in Table 3 are the same cases in the original publications.*

The data used in *Publication III* included all the data collected during Phases 1 and 2, and the data used in *Publications IV* and *V* included the data collected during the Phases 1-3. The amount of data grew along the study, which was important for understanding the phenomena observed earlier in the data analysis process, and sharpening the emerging hypotheses or theory.

3.4.3 Data analysis

All the data collected in the interviews and presentations was transcribed. The data analysis process begun with uploading the transcribed data, the text files, into a specialized analysis tool, the Atlas.ti (ATLAS.ti, 2015). The tool is used for the purpose of saving, organizing, sorting and accessing data. Managing a huge amount of data is challenging, especially when the grounded theory is applied. The tool is very useful in the data analysis process because identifying the concepts and labeling and building the relationships between the concepts is laborious and may be confusing.

During the data analysis process, I approached the data from the point of the constructivism theory (Easterbrook et al., 2008). Constructivism is a philosophical stance in which a researcher believes that knowledge is socially created and theory emerges from the data (Berger and Luckmann, 1967; Easterbrook et al., 2008). In this study the subject is software quality construction. The constructivism theory enables the researcher to interpret the phenomena (Klein and Myers, 1999), to discover the underlying meaning from the observations and suggest a theory whose validity relies upon the studied data (Myers and Klein, 2011; Strauss and Corbin, 1990). For example, in Phase 1 of the study, the theme was "*Software development, testing, quality* and *change management*". The objective of the study was "*to observe how software quality is constructed in the studied companies*". In the process of coding and categorizing the data (Strauss and Corbin, 1990), six categories were obtained: *requirements elicitation*, *development approach*, *objective of testing*, *standards used*, and *quality characteristics emphasis*. In selective coding, we studied the six categories and none of them was broad enough to describe all the categories, so we created a conceptual core category that was used to create the emerging theory. In this analysis, the core category was *quality construction of software products*.

The grounded theory suggests that the core category may be extended to a series of hypotheses and thereafter developed to a model, or a grounded theory that can explain the phenomena studied. The observations in Phase 1 suggested four hypotheses (Table 4). Our analysis in Phase 1 did not fully reach the final level of the grounded theory. The core category, '*quality construction of software products'* was not fully saturated, requiring more observations and analysis. Therefore, we continued to collect data in Phase 2, following the leads identified and at the same time trying to develop the hypotheses to a grounded theory. The leads identified in Phase 1 data analysis included: the role of management, the impact of organizational structures and communication systems, human factors in quality construction, and testing process in quality construction. The analysis of Phase 1 is presented in detail in *Publication I*.

Table 4. Hypotheses developed in Phase 1 of the study (*Publication I*)

	Hypotheses
1	It is difficult to discover an extensive set of software requirements at the beginning of the software development process.
2	Some of the quality characteristics of software products are not important to some types of software products.
3	Business requirements dictate the type of user requirements to be prioritized in the software product.
4	The quality construction of software products is constrained by budget and time (resources).

The theme of the Phase 2 study was "*Software development, testing and quality management*". The coding and categorizing process produced eight categories: *changing requirements, customer influence on quality, cooperation among the teams, developers' potentials* (creativity, innovation, experience and art), *outsourcing, method of software development, business orientation,* and *support systems.* As in Phase 1, in the Phase 2 analysis none of the categories was broad enough to describe the rest of the categories, so a conceptual category was created. Thus, the core category was *"human, the factor of software quality"*. The categories resulted from the accumulated data and observations, which stared in Phase 1 data collection and analysis. The leads identified in Phase 1 were used to modify the themes and interview questions so as to gain a better insight into the phenomena observed in the following data collection round. In the analysis process, the phenomena observed in Phase 1 were useful in interpreting the phenomena observed in Phase 2. This way, the emerging theory became more meaningful. The analysis of the Phase 2 data is presented in *Publications II and III*.

The Phase 3 data analysis included the whole volume of data accumulated from Phase 1. Generally, the analysis throughout the three phases was iterative and constant comparing of concepts throughout the process of analysis (Glaser and Strauss, 1967), where the findings were verified across the cases from the beginning to the end of the study. Two phenomena were interesting: the roles of the *human factors* and *organizational factors* in software quality construction. Pieces of the data and interpretations throughout the analysis process during Phases 1, 2 and 3 resulted in the identification of human and organizational factors that impact the software quality construction process and the quality of software products. The results of the analysis of Phase 3 are presented in detail in *Publications IV* and *V*.

Table 5 and Figure 4 show the identification of concepts in the open coding process (Table 5), and categorization and relationships between the categories in axial coding (Figure 4).

Table 5. Open coding of the transcribed interview data

Interview transcripts	Codes and categories
Yeah, that's ***essential in agile development****. This is a, let's say, vision. We design the software to go through the tests.* ***So, when we discuss, the developer says, well, I will be implementing that way. The tester says I will test it this way. Then he says no, no, it doesn't work that way. And then they start arguing and get the analyst, and then they have to agree. Okay, this will work this way, I will test it that way, so it is designed to go through the tests.*** *So, it's very important.* ***Because developer always has this positive thinking. Thinking things only the happy path,*** *gives all the estimations of the amount of work,* ***well it goes this way, yeah, two days. And then they discuss with the tester, oh!... , I have to make these exceptions and these alternative paths as well.***	Software development: Method Strategy for development: Teams (designers, programmers, testers, analysts, architects, etc.) Problem: One eye sees less details, biases in development Advantage: Many eyes sees more details, better software when teams work together

The analysis included in-case and cross-case analysis (Eisenhardt, 1989), which means that the phenomena observed in one case were compared with observations in other cases, to establish similarities or differences based on the categories. The idea is to get a deep insight into the phenomena by understanding how various factors affect the observed phenomena, for the purpose of obtaining a logical interpretation of the observed phenomena.

Figure 4 shows the relationships of the eight identified categories. Figure 4 only presents a broad picture of the interpreted phenomena of the in-case or cross-case analysis. The figure is an example of axial coding output. The aim of axial coding is to establish relationships between the categories and draw inferences from them that can deduce a core category. In the selective coding stage, the relationships are further studied to sharpen the core category for the purpose of deducing a grounded theory.

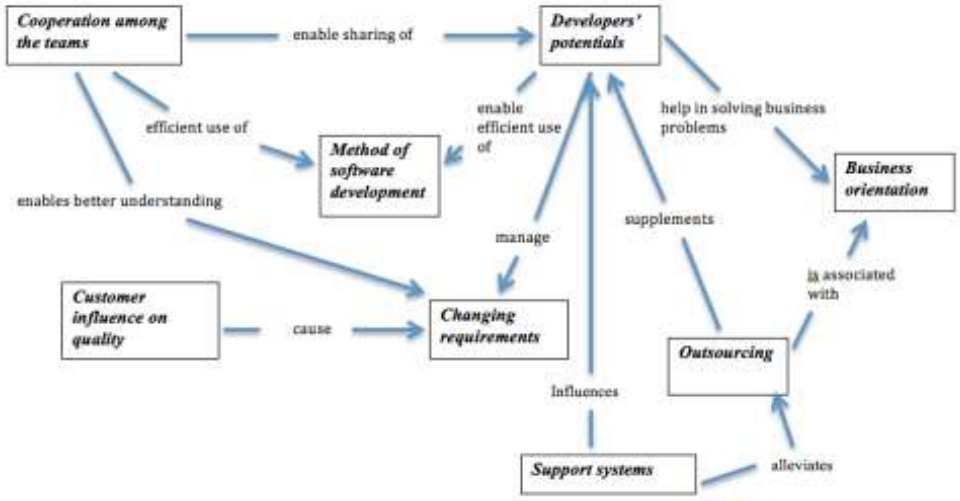

Figure 4. The output of axial coding (*Publication II*)

3.4.4 Finishing and reporting the study

At the end of each phase of the study, we wrote the peer-reviewed scientific publications that are part of this thesis (Appendix 1). Reporting of the Phase 1 study included writing one research report (*Publication I*). The report describes the role of requirements, stakeholders and resources in the software quality construction process. In reporting Phase 2, two research reports were written (*Publications II* and *III*). *Publication II* describes the role of human factors (creativity, innovation, experience and art) in software quality construction, and *Publication III* describes the challenges of software testing and its effects on software quality construction. Phase 3 included two research reports (*Publications IV* and *V*). *Publication IV* describes the management issues that influence the efforts towards the software quality construction process, and *Publication V* describes the role of the software product customer in the BYOD trend in the software quality construction process.

During the study, we built several hypotheses and presented the findings grounded on the data (*Publications I - V*). However, it is difficult to ascertain the theoretical saturation of the theories or the findings because the grounded theories are dynamic rather than static, which means that the theories may be developed further with the addition of data or change of conditions (Strauss and Corbin, 1990). At this point, it could be claimed that the saturation point was reached because the categories developed, and the findings did not change significantly with added data towards the

end of Phase 3 of the study. At the same time, there was no evidence of a case that would suggest contradicting findings, so the study was terminated because of limitation of time and fiscal resources (Eisenhardt, 1989).

3.5 Summary

Chapter 3 described the research problem and its shaping, the methodology and the research process to accomplish this thesis. The research process is summarized in Table 7.

Table 6. Summary of the research phases

Phase	**Phase 1**	**Phase 2**	**Phase 3**
Research question	What is the role of requirements, stakeholders and resources in software quality construction?	What is the role of human factors and software companies in software quality construction?	Software development, testing and quality management?
A prior constructs	ISO/IEC quality model, definitions and quality characteristics	None	None
Companies involved	11 companies, 18 interviews	7 companies, 15 interviews	5 companies, 7 interviews
Instruments and data protocols for data collection	Theme-based interviews	Theme-based interviews	Theme-based interviews
Data analysis	Grounded theory using Atlas.ti	Grounded theory using Atlas.ti	Grounded theory using Atlas.ti
Reporting	*Publication I*	*Publications II* and *III*	*Publications IV* and *V*

4 Overview of the publications

This chapter presents an overview of the research reports of this thesis. The full reports are attached in Appendix 1. These publications include one journal article (*Publication II*) and four conference papers (*Publication I, III-V*), published separately in peer-reviewed scientific conferences. For each of the publications, the objectives, results, and relation to the whole thesis are discussed.

4.1 Publication I: Software Quality Construction: Empirical Study on the Role of Requirements, Stakeholders and Resources

4.1.1 Research objectives

As software quality is not a happenstance, it is created; our first study investigated the roles of requirements, stakeholders and resources in software quality construction. The objective of this study was influenced by the standards (ISO/IEC 25010, 2011; ISO/IEC 25020, 2007), which describe software quality as satisfaction of customers through delivering software as per requirements. On the basis of the two standards, the author perceived that stakeholders, requirements and resources are involved in software development and therefore quality construction. Thus, the objective of the study was to observe and describe how software quality is constructed in software developing companies.

4.1.2 Results

In the beginning of the study, the ISO/IEC 25010 (2011) and ISO/IEC 25020 (2007) standards were used as a starting point for an a priori concept of software quality, because they both provide software quality requirements and evaluation (SQuaRE). We approached software quality construction in the context of software development through the Software Development Life Cycle (SDLC) processes. We chose to follow the quality construction process through the SDLC processes because the SDLC categorizes the software development activities into stages of development, and it is thus easy to compare a set of similar activities in the same SDLC stage across the studied companies. The findings of the research indicated that quality construction practices vary considerably across different companies. Differences were noted in the type and extent of customer involvement in the software development process, the methods used for requirements elicitation, and the objectives of software testing.

Based on the results, the following conclusions were drawn:

1) **Requirements elicitation in software development is an ongoing process even after the product has been delivered to the customer.**
   The results of this study suggest that the discovery of software requirements is an ongoing process, regardless of how extensive requirements were elicited at the beginning of the project. According to our observation, the growth of requirements can be viewed in two perspectives: discovery of new requirements and improvement of elicited requirements. The discovery of new requirements is obvious in most development projects. In many cases, the customer adds new requirements or modifies requirements presented earlier during the software development process or even after the software has been deployed in the production environment. On the other hand, the study describes 'improvement of elicited requirement' as the result of the requirement analysis process in which the requirements are prioritized and optimized. This means that same requirements are redefined and represented in a more meaningful way. It was also observed in the study that the use of requirements management tools and systematic customer involvement influences the process of requirements elicitation and analysis. In some cases, the requirements management tools were observed to have a negative effect on software quality because the developers did not communicate directly but used the system only, which resulted in different interpretations of the requirements. However, requirements management tools, systematic customer involvement and cooperation among developers have a positive effect on software quality.
2) **The criticality of the quality characteristics of the software depends on the type of software user and application domain.**

During software quality construction, the software developers focus on the customer. In this respect, the requirements are treated and presented depending on the type of user. For example, we observed that the attractiveness of the interfaces for the software developed for the company's own internal user is different from the software developed for other customers in the public. The interfaces for public software are made more attractive than the interfaces for software developed for the company's internal use. This phenomenon suggests that the level of experience and knowledge of users influences the choice of how certain requirements are addressed. On the other hand, the application domain sets the levels of priorities in the quality characteristics. For example, in the banking domain security is the top most important characteristic, while in the gaming industry usability and interfaces are highly prioritized.

3) **Business requirements dictate the type of user requirements to be prioritized in the software product.**
   The aim of software development is to address specific business and customer needs. The type of business determines the type of customers, so catering for the needs in the software should align the customer with the business requirements. Customer requirements that contradict the business requirements are not quality requirements; they should be removed. The aim of software development should be satisfying business requirements and customer requirements at the same time. However, some compromises may be made in the requirements prioritization process as a tradeoff between customer requirements, business goals and the customer segment.
4) **The quality of software is resource-dependent.**
   The quality of software is constrained with budget and time. The study uncovered a challenge for quality construction when the budget is fixed. It was observed that most projects suffer from inadequacy of funds, time for development and thorough testing of software products. For example, it was observed that governments follow a 'tender system of procurement' that requires cost estimation before the project begins. During tendering for the work, companies are forced to quote a competitive price and time for finalizing the project. As a result, more requirements are discovered during the actual development, which means more time for development and testing at the same fixed cost and time of contract. This phenomenon has contributed to poor quality of the products.

4.1.3 Relation to the whole

The results of this study identified issues that paved way for further studies of the subject. For example, we found the importance of requirements growth, which evolves in a continuous loop from the inception of the software requirements from the

customer or developer, to a maintenance period, when the developed software is being used at the customer side. This gave us the picture that the software requirements elicitation process is not only necessary to find new requirements but to find a new way of representing the same requirements better in the software. Furthermore, there should be rather a decrease of requirements represented in software than trying to address more requirements in it. Thus, there should be a kin process of prioritizing and optimizing the elicited requirements so as to address specific customer needs in a more detailed manner. Prioritization and optimization is a continuous process, which begins with requirements analysis and goes further during and after the SDLC where the necessary updates and upgrades are developed. In this regard, customer feedback is treated as potential requirements that can be used to improve the products or processes employed in the development.

The results of this research provided the leads that were followed in further research. For example, In *Publication II* we studied the role of 'support systems' such as R&D and marketing in the identification of software requirements before development and after the development process. *Publication IV* describes the effects of organizational structures in the requirements elicitation process and communication between the development teams. In *Publication V* we studied the role of the software product in the Bring Your Own Device (BYOD) trend, and exposed the issues affirming that a quality product is not necessarily a problem-free product, but one that satisfies its user.

4.2 Publication II: Software Quality Construction in 11 Companies: An Empirical Study using the Grounded Theory

4.2.1 Research objectives

The objective of this study was to investigate the role of human factors in software quality construction. The human factors were ascertained and evaluated inside the company (i.e. development, R&D and marketing functions) and outside the company (i.e. customers and consultants). The stakeholders outside the companies were studied through their involvement and/or participation in various activities reported by the practitioners in the software companies.

4.2.2 Results

The results of this study indicated that software quality construction is influenced by several factors, such as the development context, agility, outsourcing, developer's potential and support systems (i.e. technology, infrastructure and feedback system

such as R&D, marketing teams, media, etc.). The development context refers to customer context or developer context. Customer context means that the software is developed to address specific requirements as the customer suggests, and the developer context means that the software is developed to address customer requirements but as the developer desires. The developer context is applicable when the software is new to the users, and therefore the requirements emanate from the developer's ideas. In this regard, customer involvement is low. Software developed in the developer context is a result of the developer's creativity and innovation. The findings of this study suggest that the development context is influenced by customer involvement. Higher customer involvement results in the developer's shift to the customer context, which means that the software will address the customer needs better.

Cooperation between the developers was observed to be one of the success factors in software quality construction. When the developers work together they understand the requirements better and achieve better results in SDLC activities in general. For example, a software tester in one of the interviewed companies explained that when the testers are involved from the planning stage of the projects and in the development, it improves the testing work and they get better quality products. The testers also claimed that in several occasions they have been able to suggest improvements to the software during development. The developers may cooperate while applying various methods of development, however, and the results of this study suggest that agility promotes cooperation across and inside the development teams.

Outsourcing was found to be one of the important activities in quality construction. Outsourcing provides software companies necessary skills and resources, such as tools and infrastructure to achieve better quality for their products. For example, some of the studied companies hired consultants to deal with specific issues in the products they develop and other companies outsourced facilities from large companies such as Amazon and Google for the purpose of improving the reliability, performance and efficiency of their online products.

The findings of the study suggested that the developers' potentials i.e., human factors such as creativity, innovation, experience and art, are not limited by development methods but by customer demand and feedback systems. Careful thinking of what would be the customers' requirements and building artifacts and products that will address those particular needs contribute to the quality. When quality is defined from the customer's point of view, the developers are bound to customer requirements. On the other hand, the developers are limited to necessary customer information, which will directly or indirectly lead to discovery of customer requirements, and therefore the use of feedback systems (i.e. R&D, marketing, media, etc.) is essential.

Tools and infrastructure appeared to be fundamentally important in quality construction. However, the findings of this study suggest that tools do not limit developers. In this regard, different developers may use the same tools and environment but build different quality on the same type of software. Thus, human factors determine the product quality, not the tool.

4.2.3 Relation to the whole

This study was a continuation of the leads picked from the results of *Publication I*, which was about the role of requirements, stakeholders and resources in the software quality construction process. We were convinced that it is important to investigate the role of human factors in the software development process, because the human is the resource that manages and uses the other resources to add value to the processes and products. For example, the importance and challenges in requirements elicitation and prioritization are discussed in *Publications I* and *IV*. However, the developers' creativity, innovation, experience and art emerged as important human factors in the quality construction process. The results of this study indicate that the support systems, such as R&D and marketing are important in helping the developers to understand the actual needs of the customers.

4.3 Publication III: Organizational and Customer-related Challenges of Software Testing: An Empirical Study in 11 Software Companies

4.3.1 Research objectives

The aim of this empirical study was to understand organizational and customer-related challenges that affect the software testing process in software developing companies and how these challenges affect software quality construction.

4.3.2 Results

Seven main findings were presented in this study:

1) **The low involvement of testers in the planning phase of the software development life cycle (SDLC) was observed to cause underestimation of the scope of testing and resources required.**
   It was observed that late involvement of testers in projects negatively affects the estimation of the testing resources and the scope of testing. Proper estimation and definition of the scope of testing is an important basis for negotiations about the cost and time of a project, and therefore

important for contract management. Underestimation of testing resources and the underestimated scope of testing were observed to have a negative effect on the quality of products.

2) **Poor buy-in for testing from project managers.**
   Buy-in for testing from project managers led to insufficient involvement of testers in some SDLC activities. This behaviour contributed to insufficient testing of software products and therefore negatively affected the quality of the products.
3) **The risk-taking behaviour of project managers.**
   The project manager in some of the studied companies did not pay much attention to some important tests and took a risk of ignoring them. This behaviour is aggravated by shortage of time, over-trusting the development team and over-dependence on the support systems.
4) **Vertical organizational structures and modes of operation within the companies limited the mandate and visibility of testing and quality assurance units in development projects.**
   Organizational structures and mode of operations were noted to cause a significant challenge to the software testing process. Four organizational structures were described (Park et al. 2012): *vertical, horizontal, block diagona*l and *block matrix*. Testing was observed to be inefficient in the vertical organizational structure, where the functional units are efficient within themselves but not horizontally across other functional units or departments. Therefore, it was difficult to communicate across various departments, which led to department managers failing to coordinate resources and activities, which resulted in poor product quality.
5) **Oversights in budgeting and contracts between customers and the software developing companies limited the extent of testing.**
   Software development activities are often bound by contracts. Failures to estimate the scope of testing accurately and allocate sufficient funds limit the extent of testing and therefore have a negative effect on the quality of products.
6) **Poor resource management.**
   Tools, technology and infrastructure have a positive contribution to software quality. However, poor management of company resources was associated with poor use of available systems. For example, in some of the studied companies requirements management systems resulted in a communication gap between development teams, and consequently between developers, which led to a conflict in requirements interpretation and hence negatively influence the quality of products.
7) **Customers' willingness to co-operate in after-sales testing limited the extent and efficiency of testing.**

The testing of software products is extended from testing requirements and products during development to testing the products in the production environment. The extent and depth of testing help in the improvement of products. In this study it was learned that some of the customers were not willing to cooperate fully with the tester, which limited the extent of testing at the customer side, and therefore limited the possibility of the testers to discover issues for improvement in the software, which in turn negatively affected the software quality.

4.3.3 Relation to the whole

According to Dromey and McGettrick (1992), software quality must be designed into the software from the outset. This process includes identification of design principles for realizing the form, structure, strategy and process that present quality in the software, and formulation of a design process that may be employed to construct quality into the software products. In this respect, testing is an essential component of the software quality construction process, which determines how well the designs are implemented or whether the desired requirements are met. In this study, we followed the leads obtained in *Publication I*, in which we studied the role of requirements, stakeholders, and resources in the software quality construction process. Our objective was to find the issues at the customer or organization's side that affect the quality construction process. For example, the data analysis revealed several challenges of software testing, as reported in *Publication III*.

The leads concerned the role of the stakeholders in *Publication I* were followed in Phase 2 of the study, reported in *Publication II*. The emphasis in *Publication II* was on the influence of human factors within and outside software companies on software quality. The leads identified in the study of *Publication II* were investigated further in the next study to get the deeper insight into the matter. The results of *Publication III* indicated that the stakeholders inside and outside software companies variably influence the testing process. For example, the scope of testing is limited by the budget and willingness of the customer to involve the testers in product testing, especially after product delivery in the production environment. The importance of involvement of various development teams is discussed in *Publications I* and *II*. In the study presented in *Publication III*, we discovered the challenges involved in the involvement of teams in the testing process and the impact to quality. For example, the project manager's decision to ignore the involvement of the testers in the project planning stage results in underestimation of the resources required for testing, or failure to define the scope of testing, which in the end negatively affect the testing process and the product quality. The management issues influencing software quality construction were further studied and reported in *Publication IV*.

4.4 Publication IV: The Influence of Management on Software Product Quality: An Empirical Study in Software Developing Companies

4.4.1 Research objectives

The objective of this study was to understand the management issues that influence software quality construction. Management issues in this regard refer to managers' general activities and their influence on software quality.

4.4.2 Results

The study presented six findings describing managerial issues in software developing companies:

1) **Managers' technical skills, experience and knowledge in software and the domain influence the developers' productivity**.
   Managers play the 'quality championship' role, which requires them to be good in their area of leadership, so they require both good managerial and technical skills. It was found in this study that those managers who do not have software technical skills might not be able to communicate with the developers at their technical level, which may result in failures in software projects.
2) **Top-down decisions deprive transparency and affect the efficiency of requirements prioritization.**
   When managers make decisions without involving the teams (i.e. a top-down decision), especially on requirement prioritization, the following problems were observed: a communication gap between the managers and teams, wrong interpretation of requirements, low cooperation between the managers and teams, and lack of detailed information about requirements. These managerial problems negatively affect the quality of the products.
3) **Communication between managers and customers and realistic estimation of resources sustains good customer relationships, and has a positive effect on product quality.**
   Managers play an intermediary role between the customers and the company. They communicate with the customers, do negotiations and manage the tradeoffs and disputes. Good estimations of project costs and requirements enable the company to avoid problems with the customers and sustain good customer relationships, which at the same time contributes to good quality of products because the product quality is

negatively affected by resources and availability of information (*Publications I, III* and *IV*). This finding suggests that when there is good communication between software companies and their customers, it is easier to understand the customers better, and therefore it is much easier to meet their requirements.

4) **The aim of managers' decisions on resources is not to reduce expenditure but to achieve long-term goals for the organization and the customers.**
   The core mission of managers is to achieve both organizational goals and customer goals. Decisions on resources, such as training, were observed to be difficult because some training are expensive. However, managers focus beyond the immediate cost and see future benefits for both the organization and the customer. For example, investment on training benefits the customer with good quality products, and at the same time benefits the company with profit and competitive edge.
5) **The managers' choices and decisions on people variably affect other resources and the quality of the product.**
   Human factors are the resource that brings meaning to the other resources. The ability of managers to make right choices and decisions on the human and other resources has a positive impact on product quality. For example, developers using same tools and infrastructure will produce software of different quality depending on the person behind the tools, so it is up to the managers to allocate duties accordingly for better results. A wrong choice of people and allocation of resources may lead to poor quality of products.
6) **Organizational structures have an influence on the teams and product quality.**
   Software quality construction is an information-intensive process that requires efficient flow of information within the development teams, between the managers and development teams, and between customers and the project managers or development teams. Organizational structure and the channel of communication within the organization are factors that influence smooth communication and hence influence the quality of products.

4.4.3 Relation to the whole

The initial stages of the research project considered a prior concepts, quality definitions, quality models and characteristics obtained from the ISO/IEC 25010 (2011) and ISO/IEC 25020 (2007), so Phase 1 of this study investigated general software development activities in the SDLC focusing on requirements, stakeholders and resources. Several leads were identified and the research revealed several findings

reported in *Publications I, II,* and *III* from both Phases 1 and 2 of the study. For example, in Phase 1 (*Publication I*) we identified the leads related to management and requirements, resources and stakeholders. In Phase 2 we identified the leads related to the influence of human factors and the role of management (*Publication II*). Also, several challenges of testing related to management were identified (*Publication III*). At the end of Phase 2 we decided to follow the leads that we collected in Phases 1 and 2 to follow the impact of project managers on software quality, and the results were reported in *Publication IV*. In Phase 3, the themes and interview questions were modified to collect more data and we were able to get more insight into the role of software project managers in the software quality construction process. The aim of the study was not to identify what the managers should do, but rather to identify issues that affect software quality directly or indirectly.

In Phase 1 (*Publication I*) we studied the role of requirements in software quality construction. The leads identified in the data analysis in Phase 1 were taken forward to Phase 3, where deeper investigation was conducted on the requirements, and it was revealed that management processes and communication such as top-down decisions deprive transparency and have a negative effect on the process of requirement prioritization (*Publication IV*). The role of resources was discussed in *Publication I*, and in *Publication IV* the manager's role on the resources was further discussed, which revealed that the aim of managers' decisions on resources is not to reduce cost but rather to achieve long-term benefits for the company and customers at large. Testing as an important aspect of quality construction was discussed in *Publication III* together with the role of managers that affects the testing process negatively. *Publication IV* further discussed the role of managers and the organizational structures that variably affect the efficiency of the teams, including testers, in achieving their goals, which impacts the product quality.

4.5 Publication V: Role of Software Product Customer in the Bring Your Own Device (BYOD) Trend: Empirical Observations on Software Quality Construction

4.5.1 Research objectives

This empirical study investigated practical experiences in software developing companies, and the objective was to understand the challenges in meeting the quality requirements of software products in a Bring Your Own Device (BYOD) environment. The BYOD trend is very important because companies or businesses, among other benefits, are able to exploit their workers' personal devices to extend their productivity and availability beyond working hours. However, there are many

security risks involved in the BYOD trend. The BYOD trend is rampant and popular, but no device comes without software, so in this study we looked at the Bring Your Own Software instead of the devices.

4.5.2 Results

This study presented seven findings:

1) **Visible features of the software and functional requirements supersede nonfunctional (quality) characteristics when dealing with customer requirements.**
   It was observed that customers evaluate software or systems based on visible features or behaviours such as interfaces, usability, outputs and efficiency when the software or system is being used (external software quality). In this respect, in the BYOD environment, external quality is very important because the purchase decision depends on how the customer is attracted to by the product.
2) **Quality depends more on the market decision than standards' requirements.**
   Technology is changing at a very high pace, likewise the requirements. Software companies are not bound to standards, but customer requirements. People buy what they see other people buying, so companies also focus more on what people like than what is described in the standards.
3) **Companies focus on 'just enough quality' and not on 'high quality' products.**
   Customer satisfaction relies rather on "how the product *attracts* the customer" than "how *good* the product is". In common circumstances, companies deliver *just enough quality* rather than high quality products as long as the customer is satisfied. It was observed that there is a risk of developing products beyond given requirements and specification because of incompatibility with other systems at the customer side. What works best for the customer is the "high quality" product for them and not what the developers think is too good, because quality is referred to as "what satisfies the customer".
4) **Software quality has a dimension of cost.**
   Customer satisfaction includes being satisfied with the cost of products. Two important questions to answer before launching software development projects are: *What is the important quality to build for a particular customer?*, and *How to build such quality into a product at a reasonable cost and time?* The first question focuses on the customer' requirements and the second focuses on the company's capability to deliver the product as the customer requires. When the customer is not willing to pay for higher quality, then he chooses to buy a

rather cheaper product and may well be satisfied though the quality is not as high.

5) **The quality aspect of policy, i.e. organizations try to alleviate threats brought by employees' devices or software through policies.**
   As the BYOD trend allows the users to bring their own devices to their working places according to their discretion, it is difficult to standardize the quality requirements for those devices and the software running in them. In this respect, policies may be regarded as important characteristic that will determine the quality of the user devices. The policies in the user devices should be flexible to conform to the requirements of the user's working place information systems.
6) **Simplicity and attractiveness of devices sell poor quality software.**
   Usability is the key item for software acceptance or rejection (Chao, 2009). In the BYOD trend, it is the user who decides the type of the device to buy, so simplicity of the device and the interfaces are the selling points of software regardless the actual quality of the product.
7) **The number of product features does not affect the sense of quality, but quality characteristics do.**
   In this study it was observed that the quality of products is not much affected by the number of features, but by functionality and usability. Users choose only a few features to use and forget about the rest. On the basis of this observation we can also argue that product features have positive impact on product quality in similar manner as quality characteristics. In most devices, the features are conspicuous and easy to count or compare between devices from the layman's point of view, but quality characteristics are difficult to see and evaluate. Therefore, devices with attractive features are likely to cause more threat in the BYOD environment because users are likely to buy them despite of internal weak characteristics.

4.5.3 Relation to the whole

According to ISO/IEC 25020 (2007), quality is defined as meeting customer requirements. *Publication I* contained definitions of quality, and the findings discussed the role of requirements, customers and resources. In Phase 3 of the study, we investigated the role of customers (i.e. stakeholders) in the BYOD environment in software quality construction. The results of this study were presented in *Publication V*. The leads gathered in Phases 1 and 2 were examined during data collection and analysis, and results were reported in P*ublication I*. The results in *Publication I* indicated that the quality of the software depends on the type of software, user, and application domain. The results in *Publication II* suggested that the art and creativity invested in the product add attractiveness and sense of quality to the product, which increases customer satisfaction. In Phase 3, the study reported in *Publication V* showed

that functional requirements supersede the nonfunctional (quality) requirements, which means that customers are attracted by the visible features of a software or device, and make the buying decision based those choices, including the product price, regardless of how good the product may be.

The results reported in *Publication V* suggested that companies focus on 'just enough quality' and not on 'high quality' products. This result concurs with the result reported in the *Publication I,* which indicates that a quality software product is not necessarily technically a problem-free product but it must satisfy customer needs. Figure 5 summarizes the findings and relation to the whole.

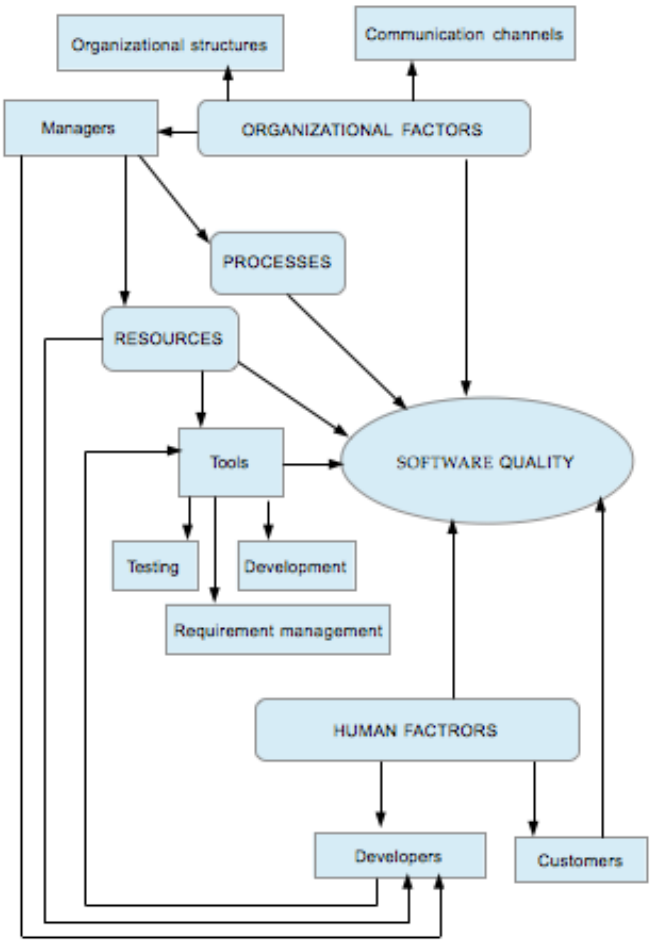

Figure 5. The human and organizational factors in software quality construction

Software quality construction is a socio-technical process that can be influenced by

several factors (Hovenden et al., 1996). The scope of this thesis is confined to the factors surrounding humans (i.e. developers and customers), the human factors (*Publication II*) and software companies, the organizational factors (*Publication III* and *IV*). In Figure 5, four major factors (in caps) are presented: human factors, resources, processes and organizational factors, and other sub-factors are presented in rectangles.

The arrows indicate the direction of influence or where the entity belongs. For example, customers have direct influence on product quality (*Publication III* and *IV*); tools have direct influence on the quality of testing, development and requirement management (*Publication I, III* and *IV*), etc. Resources include developers, tools, etc.; organizational factors include organizational structures, channels of communication and managers (*Publication I, II, III* and *IV*).

According to Park et al. (2012), organizational structures have an impact on productivity of the functional teams within organization hence variably affect quality of the software products (*Publication III and IV*). Communication channels, skill and experience of project managers are other factors that impact the quality of products. For example, Suominen and Mäkinen (2013) argue that the success of projects depends on the management of processes that are domain specific at business level. In Figure 5, managers manage resources and processes (Boehm and Ross, 1989; Colomo-Palacios et al., 2013) (*Publication IV*).

Despite tools and processes, humans (i.e. developers and customers) play a primary and fundaments role in quality construction (Petre, 2010; Boden, 2004; Shneiderman, 2007) (*Publication I*). On the other hand, in the BYOD trend, products are developed focusing on individual end-user (Sangroha and Gupta, 2014; Scarfo, 2012) so the customer directly influences the quality of the products (*Publication V*).

4.6 The author's role in the joint publications

During Phase 1, the initial design of the study, a research team involving five persons, including the author prepared the data collection tools jointly. The author, in collaboration with two other researchers, collected the data. The author analyzed the data and wrote major parts of *Publication I* as the first author.

During Phase 2, the author prepared the data collection tools in collaboration with another researcher in the team who was also involved in the data collection. The data analysis and reporting were done separately because of different reporting purposes. The author analyzed the data and wrote major parts of *Publication II* and *Publication III* as the first author.

During Phase 3, the research team sharpened the data collection tools. The author collected and analyzed the data and wrote major parts of *Publication IV* and *Publication V* as the first author.

5 Research contribution, implications and limitations

This chapter presents the contribution, implications and limitations of the study. The contribution section includes summary of the findings of the three phases of the research. The research implications section describes the implications for practitioners and researchers. Finally, the research limitations and validity are discussed.

5.1 Research contribution

This research focused on software quality construction by examining software development activities in various software companies. The a priori concepts and definitions applied at the beginning of this research were adopted from the ISO/IEC 25010 (2011) and ISO/IEC 25020 (2007). The Software Development Life Cycle (SDLC) was used as a guide to identify and categorize the software development activities. During Phase 1 of the research we studied the role of requirements, stakeholders and resources in software quality construction. The findings indicated that requirements elicitation is a continuous process in which requirement prioritization is described in a pyramid form, so that the growth of requirements should decrease in size, from general business and customer requirements to software requirements, characteristics and product features (Figure 1). The process of prioritization should be able to determine the high value requirements and optimize them by redefining and refining them to address the specific customer requirements better. During Phase 1, we were able to note that the requirements can be both '*told*' and '*untold*'. The untold requirements are latent requirements which cannot be elicited because of human

limitations of expression, ignorance or misinterpretation of ideas or needs. Thus, the *untold* requirements are not directly expressed and may not be elicited during the requirements elicitation process. The *told* requirements include requirements which are identified and directly elicited from customer expressions, phenomena or situation on the customer side.

While the ISO/IEC 25020 (2007) definition of software quality recognizes the customer as a key person whose requirements should be met, this research suggested that the objective of software quality construction is broader than meeting costumer requirements. There are organizational goals that should be expressed in the software products for the purpose of keeping the brand in the market and for gaining a competitive edge. In this respect, this thesis uncovered the role of human factors (creativity, innovation, experience and art) in software quality construction. For example, accommodating the *untold* requirements requires developers' art, creativity and experience in the domain to address the requirements and thus develop software that caters for such needs for present or future requirements. *Publication II* elaborated this contribution in closer detail.

This research defined software quality construction as an *information intensive process*, whose success depends on the amount and clarity of information involved in the software development process. Any deprivation of information within software companies, within development teams or between customers and developers may lead to quality issues. For example, in *Publications III* and *IV* the findings suggested that the organizational structures, mode of operation and information flow within the software companies may prevent transparency and information flow, and therefore have variable impact on the quality of the products.

The author perceives that quality is often defined based on the customer's view and not on the developer's view, which means that what the customer perceives as high quality is a high quality product to the customer regardless of how well the product is built. On the other hand, the 'cost dimension' of quality emerged as an important attribute that contributes to customer satisfaction. Furthermore, the product was evaluated based on compatibility with existing systems at the customer side, which means the products developed beyond specifications were observed to bring problems and not considered as of high quality. Thus, the quality goal for software companies is not to deliver 'high quality' but 'just enough quality' according to the customer's requirements and existing systems at the customer side. Thus, the success of the quality construction process depends on the skills and knowledge of the software developer, the customer as a person and as a business entity, and the knowledge of the domain.

In the BYOD trend, the software companies reorient software development to individual customers and not to the area of application of the software. This trend has

led to serious security risks at workplaces where individuals use personal software applications installed in their personal devices. The findings of the interpretative study conducted during Phase 3 suggested a '*policy aspect of software quality'*, which means that the software should be flexible and able to adapt to corporate security policies. Flexibility is needed for the purpose of increasing the global value of the software rather than optimizing on the individual customers only. We described this as meeting a '*communal quality goal*', i.e. individual and business goals of software quality. This conclusion is a deduction from the hypotheses presented in *Publication I*, which described the goal of requirements elicitation as meeting both customer and business goals in a jig-saw-fit manner (Figure 1), so that the software product should include both the customer and the business goals. Customers may simply interpret the unidentified or wrongly interpreted requirements as 'poor quality'.

Finally, the managers of software development projects at different capacities are described as '*quality champions'*. Good developers and tools are critical in software quality construction. Nonetheless, the managers who manage resources and processes influence the productivity of the developers and other resources and hence influence product quality. A manager who does not have the necessary skills and knowledge of the software development domain and the business environment of the customers, may not be able to lead the teams to deliver quality products. This phenomenon was observed in Phase 3 of the research, and described in detail in *Publication IV*.

5.2 Implications of the research

This section presents the theoretical and practical implications of the study. The theoretical implications are important for the academia for conducting further research. The practical implications give advice to software practitioners.

5.2.1 Theoretical implications

This study focused on software development activities in order to understand how software quality is constructed into software products. The results reported in *Publications I-V* indicated that software quality is a result of many factors that emanate from disciplines beyond computer science. The importance of customer involvement, communication and collaboration between development teams, the qualities of project managers, tools and processes were obvious observations in many companies. However, further investigation is required to understand how these factors impact product quality. For example, organizational structures, mode of operation and communication channels emerged as factors that affect product quality.

Software quality like any other quality is difficult to define and measure. There is no standard objective measurement for quality (Jørgensen, 1999). So, it is difficult to estimate how much quality is implemented, except from software testing results and customer feedback, which also vary widely depending on the type and knowledge level of the customers. Other factors such as price, policy adaptability (*Publication V*) and compatibility of the software to existing systems at the customer side seem to be outside the product itself, yet have direct influence on customer satisfaction and hence command different levels of judgment on the product quality. For example, a customer may reject a well-built piece of software because it is too expensive or causes problems to the exiting systems, and accept another software of lower quality because it is cheap or it works well with the existing systems. Thus, the focus of quality construction is to meet 'communal' objectives. The results of this study suggest that more research is required in order to understand the quality requirements and suggest better methodologies to identify and elicit the requirements and improve the quality definitions.

Human factors emerged in this study as the center of software quality construction. Four traits were discussed: *creativity, innovation, experience* and *art*. Although many studies define quality as meeting customer requirements, some describe "created quality". For example, how to deal with the 'untold' requirements, the requirements that are not elicited in the process of requirements elicitation, yet are important customer needs? How to design, build and test such software whose requirements are 'untold'? Creativity, innovation, experience and art in software design, development and testing enable the developers to represent the needs in the product in a way that will satisfy the customer when the software is used. In the testing process it is critical to define the scope of testing and select the items for testing, including the environment. Creativity, experience and art of testing will lead to different results, which lead to different product quality. Thus, the findings of this research recognize the quality that emanates from the developer's creativity, experience in the field, use of innovation and artistic approach to SDLC activities and suggest more research to investigate the role of human factors in quality creation.

5.2.2 Summary

The following are the key implications to research of this thesis:

- The study suggests further research on the impact of organizational structures and mode of operation on the software quality construction process.
- The objective of quality construction is to meet common goals which include satisfaction of customers and meet organizational goals through the same software products. More research is required in order to understand the 'communal quality requirements' and suggest better methodologies to

identify and elicit the communal requirements and improve the quality definitions.

- Research is required on the roles of human factors on quality construction, specifically on the created quality (i.e. the quality that emanates from developers creativity) in order to understand the methods of requirements elicitation and how the customers are involved in such kind of development.

5.2.3 Practical implications

Software quality is the top agenda item for software practitioners. Poor quality has cost and/or legal implications, and therefore quality is not an option but a necessity. Software companies should satisfy the customers not only for the purpose of achieving quality goals, but also to compete with rivals and make profit. In the business perspective, quality construction is a strategy which aims at meeting both customers' requirements by satisfying their needs and the company's goals by making profit and avoiding losses such as penalties resulting from sub-standard products or breach of contracts (*Publication III*). For example, in *Publication III*, the involvement of testers in the early stages of development were observed to be very important for cost estimation, determining the scope of testing and understanding the customer requirements. Generally, the results in all the three phases of the study indicated that the involvement of all development teams in all stages of SDLC contributed to software quality.

Software companies need to invest on infrastructure, tools and processes, but the key of success in quality construction is the 'quality of people' i.e. the human factors involved in the SDLC. For example, it was revealed in one of the studied companies that eighty percent (80%) of software quality issues were caused by twenty percent (20%) of the developers. This observation suggests that if the project manager is able to identify the twenty percent of the developers who cause problems and keep them away from coding, will solve the quality problems to a certain extent. So, the duty of the project manager is to identify the potentials of the people in his/her team and allocate tasks accordingly, or suggest necessary training for better results. Although this observation cannot be taken as a theory, it gives an indication that managers' ability to understand the potentials of their teams, and allocate duties to the teams carefully, may result in better quality of the products.

This thesis has described the software quality construction process as an 'information-intensive process'. Success depends on the clarity of quality objectives, mutual understanding of requirements between developers and customers, and easy flow of information within various development teams. One of the factors that determine the flow of information within software companies and influence the software quality construction process is the organizational structure (*Publications III* and *IV*). On the

other hand, managers play an intermediary role in communicating with customers on behalf of the company. The efficiency of managers in communicating builds good customer relationships and at the same time enables the development teams to be furnished with important information about the customers.

5.2.4 Summary

The following are the key implications of this thesis to practitioners:

- The findings of this study indicate that the involvement of all development teams at all stages of SDLC contribute to better software quality. Software companies should be able to involve all the development teams or their representatives in all stages of SDLC for the purpose of understanding the requirements, quality goals and proper estimation of resources required for the projects.
- The quality of tools, infrastructure and technology is important in quality construction, but the quality of people determines the quality of products. Project managers should be able to identify the right people and allocate duties and suggest training as appropriate.
- The flow of information during development contributes to good product quality. The channels of communication should be as clear as possible; information flow within the organization and between the organization and the customers should be efficient.

5.3 Validity and limitations of the thesis

There are several points where the validity of this thesis can be questioned. The grounded theory method used for data collection and analysis aims at creating theory from the data (Strauss and Corbin 1990). However, the theories created through the grounded theory are not static, they can change with the addition of data. To address this issue, we collected the data until the saturation point was reached. The saturation point is the state whereby the data collected seem to add no new information about the studied phenomenon (Glaser and Strauss 1967; Strauss and Corbin 1990). For example, in Phase 1 some companies required fewer interviews than others because the concepts started repeating during the interviews, while in other companies new concepts emerged and necessitated more interview sessions (Table 3). Similarly, during Phases 2 and 3 data collection was conducted until the saturation point was reached.

This study was done mainly in one country, which poses a territorial bias. However, several companies involved in the data collection were large international companies with branches in many countries (Table 3). So we believe that although the research was done mainly in one country, the experience of the interviewees in international companies represent viewpoints that are applicable everywhere.

Robson (2002) discusses three threats to validity: *reactivity* (i.e. interference of the researcher's presence), *researcher's bias*, and *respondent's bias*. The main data collection method employed in the study was interviews. To reduce the threat of the researcher's interference, we allowed the interviewees to answer the questions without further guidance. In many instances, the interviewees delivered vital information beyond the asked questions, which revealed leads for further investigation. We used the tape-recorder for smooth and efficient capturing of the information without interrupting the interviewees. Furthermore, we avoided surprising the interviewees with the questions by sending the interview questions to them beforehand.

The author conducted the research with a team of researchers, so we believe that the researcher's bias (Robson, 2002) was minimized. For example, at the beginning of Phase 1, five researchers, including the author, prepared the data collection tools, and three members of the team collected the data. During Phase 2, the data collection involved one member of the team. However, the author, in collaboration with two other members of the team, prepared the tool for data collection. In Phase 3 the author prepared the data collection tools in collaboration with two other research team members. The author conducted the data analysis in all the three phases, but the research project manager and two other team members discussed the results during the writing of the reports (*Publications 1-V*).

To address the respondent's bias (Robson, 2002) we interviewed the key persons who were involved in the actual software development processes in their natural environment in software companies. Klein and Myers (1999, p. 71) discuss the principle of hermeneutic circle, which claims "all human understanding is achieved by iterating between considering the independent meaning of parts and the whole that they form". The concepts extracted in the data analysis were compared across the individual companies and between companies to find the underlying meaning and validity. Thus, the deductions drawn from this research are accompanied with quotes from the data obtained from different studied companies, explaining similar phenomena.

Maxwell (1992) discusses five threats of validity to qualitative research: *descriptive validity, interpretative validity, theoretical validity, generalizability,* and *evaluative validity.* Descriptive validity is concerned with the factual accuracy of the accounts of the data gathered in the field so that the data is not distorted. We addressed this threat by using tape-recording that captured all the data, and we hired professional transcribers

to convert the voice data into text for easy analysis. Some of the interviewees used Finnish words, and a Finnish native researcher who was part of the research team helped to obtain proper translation. We believe that the data was neither altered nor distorted. However, it was difficult to capture the gestures and physical expressions that could be meaningful to emphasize what the interviewees expressed.

According to Maxwell (1992), interpretative validity is concerned with making sense of the objects, events, and behaviors in the settings of the study for the purpose of drawing meaningful findings from the observed phenomena or scenarios. The grounded research method (Strauss and Corbin, 1990) applied provides a systematic way of analyzing data, which leads to formulation of hypothesis or a theory that is grounded on the data studied. Furthermore, in this study we compared observations across the studied cases, so I believe that the conclusions made from the data are valid.

Theoretical validity (Maxwell, 1992) is concerned with the theoretical construction from the physical and mental phenomena studied that goes beyond concrete description and interpretation. According to Maxwell, theoretical validity refers to the validity of an account as a theory of some phenomenon. In this study, the drawn conclusions were accompanied with quotes from the data, which give evidence that the findings are grounded on the data. In several instances, the same finding was grounded with data collected from various sources; therefore I believe that this threat was well addressed.

Generalizability validity (Maxwell, 1992) refers to the extent to which one can extend the account of a particular situation or population from the one directly studied. As this study was qualitative, the theories created were bound to the studied environment and may not be valid in other environments (Maxwell, ibid.). Adding to this threat, the grounded theory (Strauss and Corbin, 1990) employed produces theories that may change upon addition of data. To address this validity, the phenomena observed and categorized during the data analysis were compared first within and across the studied cases; second, the data collection process was continued until the saturation point was reached, i.e. where no new information was obtained about the phenomenon under study (Glaser and Strauss 1967; Strauss and Corbin 1990). I believe that this threat was adequately addressed.

Evaluative validity (Maxwell, 1992) refers to evaluation given by a researcher after a study. Maxwell argues that it is difficult to evaluate critically some observations especially in qualitative research. For example, researchers may avoid declaring whether an event is legitimate or not but present observations. To address this validity, the findings of this study were grounded on the data and a few deductions were made by the author when trying to build a big picture of the studied phenomena. For example the author made the assertion of a few evaluative statements such as “software quality construction is an information –intensive process”,

"software quality construction is a social-technical process", etc. However, the deduced statements were supported by the literature. So, I believe evaluative validity to have been adequately addressed.

Although we employed the grounded theory, which aims at creating theory, not all the studies reported in *Publications I-V* produced a theory. However, Strauss and Corbin (1998), and Corbin and Strauss (2008) acknowledge that not all studies will produce a theory. According to the Straussian grounded theory, the broader goal of the grounded theory is beyond theory creation. Some studies will result in important descriptions, thus failure to build a grounded theory does not make the method or the results invalid (Cooney, 2009).

6 Conclusions

This study consists of three phases, of which several observations and findings were reported in the chapter 5 above and in closer detail in *Publications I-V* (Appendix 1). The findings were obtained from continuous observations made from data collected throughout the three phases, each phase providing leads that were used for the next step of the study. Each of the research phases studied the activities within the organizations that contribute to or negatively affect the efforts towards software quality construction.

6.1 Contribution and summary

The results presented in this thesis were empirically obtained in software companies. The aim of the study was to investigate software development activities to understand the software quality construction process. The data was collected from practitioners in their natural setting. The research was divided into three phases. During the first phase we investigated the general software development activities by investigating the role of requirements, software stakeholders and resources. During the second phase we investigated the role of human factors and software companies in quality construction. During the third phase we investigated the organizational factors that affect the software quality construction process, and the role of the customers in the BYOD trend, trying to uncover the challenges involved in quality construction.

Numerous phenomena were observed in the SDLC activities related to quality construction, and the concepts were shaped as we collected more data and studied the

phenomena. The following list provides recommendations for both researchers and practitioners:

- The goal for quality construction should be '*communal*', i.e. the goals should include customers' and software company's goals for the purpose of satisfying the customers and at the same time achieve organizational goals.
- Software quality is an *information-intensive* process, so there must be as much transparency as possible.
- Software quality construction is a nonlinear process in which the identification of requirements in beyond the SDLC activities. Customer feedback is vital information that can be considered as actual requirements that can be used to improve software design, test cases and other SDLC activities.
- The development context is influenced by customer involvement. Customer involvement improves understanding of the context and addresses the requirements better.
- The development methods do not determine the level of cooperation among the development teams, but the organizational setting, including the managers' discretion does.
- Outsourcing is important in software quality construction.
- Creativity, innovation, experience and art are not limited by development methods but by customer demand and feedback systems.
- The advancement of tools and methods positively influences the quality of products despite the developer's potentials.
- Challenges of testing that impede software quality construction process:
  - Project managers give little attention to testing: poor involvement of testers in the SDLC was observed especially during the initial stages of a project, such as planning. This resulted in underestimation of the testing budged and the scope of testing.
  - Risk-taking behaviour of project managers by skipping some of the tests or allocating little time for testing. Over-trusting development teams and support systems available for the project aggravate this behaviour.
  - Organizational structures and mode of operation within software companies contribute to lack of transparency, cooperation among teams and sufficient involvements of testers in projects.
  - The scope of testing is limited by project contracts between the customer and the software company, testers' knowledge of the software domain, and understanding of business processes.
  - The cooperation of management of various departments within the company has positive influence one the efficiency of testing.

  - Testing is a resource-intensive process, so success does not depend on the technical teams only, the managers at various levels should come together and avail resources to achieve the quality goals.
  - Customers' willingness to involve the testers at the production environment limits the testing process. The limited involvement of testers in the production environment narrows down the scope of testing.
- Managers' roles in software quality construction:
  - Managers' technical skills, experience and knowledge in the software and the domain positively influence the developers' productivity.
  - Top-down decisions deprive transparency and negatively affect the efficiency of the requirements prioritization process negatively.
  - The efficiency of communication between managers and customers and realistic estimation of resources sustain a good customer relationship and has a positive effect on product quality.
  - The aim of managers' decisions on resources is not to reduce expenditure but to achieve long-term goals for both the organizations and the customers.
  - The managers' choices and decisions on people variably affect other areas of resources and the quality of the products.
- General observations in quality construction in the BYOD environment:
  - Visible features of software and functional requirements supersede nonfunctional (quality) characteristics when dealing with customer requirements.
  - Quality depends more on the market decision than the requirements of standards.
  - Companies focus on '*just enough quality'* and not on '*high quality*' products.
  - Software quality has a dimension of cost.
  - Quality aspect of policy in the software. The capability of the software to adjust to various corporate policies may be regarded as a quality characteristic in the BYOD environment.
  - Simplicity and attractiveness of devices sell poor quality software. Customers prefer devices to software; so the buying decision may be influenced by the hardware, not the software.
  - The number of product 'features' does not affect the sense of quality, but quality 'characteristics' do.

In summary, this study has established that software quality construction includes activities beyond the SDLC. The knowledge required in the quality construction is beyond the computer science discipline. For example, understanding business cases and the domain is one of the major issues in requirements elicitation and prioritization.

Organizational systems within the companies played a very important role in the quality construction process. For example, organizational structures, the mode of operation and communication channels, within the organization contribute to success or failure in the software quality construction process.

Tools, processes, methods and infrastructure are the keys in the software development process. However, the developers' potential determines the difference between the products developed in the same environment and with similar resources, so this study has affirmed that the human factor is the basis of software quality.

6.2 Future research topics

The results of this study may be extended to increase the understanding of the software quality construction process.

"Quality construction" simply means the process of developing software that satisfies the customer needs, i.e. implementing quality into software products. This thesis has presented the importance of customer involvement and cooperation between the development teams for the purpose of understanding the requirements and implementing them in the software. However, in all the studied companies, there were no frameworks that guided the practitioners in the customer involvement and requirements prioritization. Therefore, it could be useful to study to how companies could systematically involve their customers in the development process and if possible to establish a model or a framework that practitioners could use in the software quality construction.

The discovery of requirements is one of the success factors in quality construction. The study suggested two categories of requirements: *untold* and *told* requirements. It is difficult to ascertain and elicit those requirements which are 'untold', i.e. requirements that have never been mentioned or expressed. This thesis has described the role of support systems, i.e. R&D and marketing teams, which are potentially important in the requirements elicitation and quality construction process. It would be an interesting topic to study how the support teams, such as R&D, marketing teams and others could cooperate with the development teams in order to achieve better software quality.

Another important research topic could be to investigate further the role of the human factors, such as creativity, innovation, experience and art, and suggest a framework that companies could use to identify and capitalize on the human factors for the purpose of increasing productivity and the quality of products.

This thesis has presented challenges of organizational structures, mode of operation and information flow within software companies. It would be useful to carry out more investigation on the effects of organizational structures and systems or business models in the software quality construction process. Lastly, I suggest a study that would re-visit the quality requirements and characteristics and estimate the effect of pricing and policies on them.

References

Akbar, A. A., 2003. "Pay-per-use concept in healthcare: a grounded theory perspective, Proceedings of the 36th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA.

Ambler, S. W., 2009. https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/art_versus_science?lang=en, [accessed on 10th January 2015, 16:46 hours]

Arden, R., Chavez, R. S., Grazioplene, R. & Jung, R. E., 2010. Neuroimaging creativity: A psychometric view. *Behavioral Brain Research*, 214(2), 143–156.

Arshad, Y., Ahlan, A. R. & Ibrahim, S. N. S., 2013. 'Combining Grounded Theory and Case Study Methods in IT Outsourcing Study'. *Journal of information systems research and innovation*, 4(2013), 84-93.

Aslhford, N. A., Ayer, C. & Stone, R. F., 1985. *Using Regulation to Change the Market for Innovation*, http://dspace.mit.edu/bitstream/handle/1721.1/1555/%252319.PDF?sequence=1. [Accessed on 7.5.2014]

ATLAS.ti, 2015. *Qualitative Data Analysis*. Available at http://atlasti.com [Accessed on 11 January 2015]

Barney, S. & Wohlin, C., 2009. Software Product Quality: Ensuring a Common Goal, International Conference on Software Process Proceedings, (ICSP 2009) Vancouver, Canada, Volume 5543, pp. 256-267.

Berger, P. & Luckmann, T., 1967. *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*, Anchor.

Boden, M., 2004. The Creative Mind: Myths and Mechanisms, 2nd Ed. Routledge, London.

Boehm, B, W., 1984. Software engineering economics. *IEEE Transactions on Software Engineering*, Se-10(1), 4-21.

Boehm, B. W. & Ross, R., 1989. Theory-W software project management principles and examples, *IEEE Transactions on Software Engineering*, 15(7), 902–916.

Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., Macleod, G. J. & Merrit, M., 1978. *Characteristics of Software Quality*. North-Holland, Amsterdam.

Bryant, A., 2002. Grounding systems research: re-establishing grounded theory, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on System Sciences, Hawaii, pp. 3446 – 3455.

Busch, M., Koch, N. & Wirsing, M., 2014. Evaluation of Engineering Approaches in the Secure Software Development Life Cycle, Engineering Secure Future Internet Services and Systems Lecture Notes in Computer Science Vol. 8431, pp 234-265.

Chao, G., 2009. The usability test methods and design principles in the human-computer interface design, 2nd IEEE International Conference on Human-computer interaction: Computer Science and Information Technology, ICCSIT, Beijing, Pp. 283 – 285.

Charmaz, K., 2010. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*, Sage Publications.

Chotisarn, N. & Prompoon, N., 2013. Forecasting software damage rate from cognitive bias in software requirements gathering and specification process, 2013 International Conference on Information Science and Technology (ICIST), Yangzhou, pp. 951 – 956.

Christoffer, B. & Furuknap, T., 2009. Intermission: The mentality of a SharePoint developer. Building the SharePoint User Experience, pp. 229-232.

CMMI for Development, Version 1.3, 2010. *Improving processes for developing better products and services*, Carnegie Mellon University, Hanscom AFB, MA 01731-2100.

Cockburn, A. & Highsmith, J., 2001. Agile software development: The people factor. *Computer*, 34(11), 131-133.

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta P., José F., Peñalvo G. & Tovar, E., 2013. Project managers in global software development teams: a study of the effects on productivity and performance, Software Quality Journal, DOI: 10.1007/s11219-012-9191-x, Springer US.

Cooney, A., 2009. *Choosing Between Glaser and Strauss – An Example*, https://www.rcn.org.uk/__data/assets/pdf_file/0007/253249/2009_RCN_research_8.7.1.pdf[Accessed on 20.11.2014].

Corbin, J. & Strauss, A. L., 2008. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. 3rd edition. Sage.

Crosby, P. B., 1979. *Quality is Free*. New York: McGraw-Hill.

Cross, N., 2001. Designerly Ways of Knowing: Design Discipline vs. Design Science. *Design Issues* 17 (3), pp. 49–55.

D'Aniello, A., Masone, A. & Tammaro, A., 2006. Technological innovation within EDS Italia software: Experience Report. Proceedings of the Conference on Software Maintenance and Reengineering (CSMR2006), Bari, March, IEEE, pp. 355 – 366.

Dal Forno, A. & Merlone, U., 2012. Grounded Theory based agents, Proceedings of the 2012 Winter Simulation Conference (WSC), Berlin, pp.1-11.

Dieste, O., Juristo, N. & Shull, F., 2008. Understanding the Customer: What Do We Know about Requirements Elicitation? *IEEE Software*, 25(2), 11 – 13.

Dromey, R. G. & McGettrick, A. D., 1992. "On specifying software quality", *Software Quality Journal*, 1(1), 45-74.

Easterbrook, S., Singer, J., Storey, M. A. & Damian, D., 2008. Selecting Empirical Methods for Software Engineering Research. In F. Shull, J. Singer & D.I.K Sjoberg, eds, Guide to Advanced Empirical Software Engineering. London: Springer-Verlag. pp. 285-311.

Edison, H., Ali, N. & Torkar, R., 2013. Towards innovation measurement in the software industry. *The Journal of Systems and Software* 86(2013), 1390–1407.

Eisenhardt, K. M., 1989. Building theories from case study research, *Academy of Management Review*, 14(4), 532-550.

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

Fagerholm, F. & Münch, J., 2012. Developer experience: Concept and definition. 2012 International Conference on Software and System Process (ICSSP), Zurich, Switzerland, IEEE, pp. 73 – 77.

Garvin, D. A., 1984. "What Does "Product Quality" Really Mean?" *Sloan Management Review*, 4(1984), 25-43.

Gheorghiu, G., 2005. "*Agile Testing: Did anybody say webscale?*" http://agiletesting.blogspot.com/2005/02/performance-vs-load-vs-stress-testing.html [Accessed on 10.10.2014].

Glaser, B. & Strauss, A. L., 1967. *The discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine, Chicago.

Glaser, B., 1992. *Basics of grounded theory analysis*. Mill Valley, CA: Sociology Press.

Gokhale, S. S., Lyu, M. R. & Trivedi, K. S., 2006. Incorporating fault debugging activities into software reliability models: a simulation approach, *IEEE Transactions on Reliability* 55(2), 281 – 292.

Hartung, A., 2012. *Top 20 R&D Spenders - Not Good Investments*, http://www.forbes.com/sites/adamhartung/2012/11/05/top-20-rd-spenders-not-good-investments/ [Accesses on 29.10.2014]

Hirschheim, R.A., 1985. *Information Systems Epistemology: an historical prospective* in R. H. E Mumford, G Fitzgerald, T Wood-Harper (ed.), Research Methods in Information Systems, North-Holland, Amsterdam.

Hovenden, F. M., Walker, S. D., Sharp, H. C. & Woodman, M., 1996. Building quality into scientific software, Software Quality Journal, 5(1), 25-32.

ISO/IEC 15504-5:2012, 2012. *Information technology, Process assessment, Part 5: An exemplar software life cycle process assessment model.*

ISO/IEC 25010., 2007. *Software Engineering – Software Product Requirements and Evaluation. (SQuaRE) - Measurement reference model and guide.*

ISO/IEC 25020:2007., 2007. *Software Engineering – Software Product Requirements and Evaluation, (SQuaRE)-Measurement reference model and guide.*

ISO/IEC TR 9126-1:2001, 2001. *Software engineering – product quality – Part 1: Quality model.*

ISO/IEC/IEEE 24765:2010, 2010. *Systems and software engineering – vocabulary.*

Issac, G., Rajendran, C. & Anantharaman, R. N., 2006. An instrument for the measurement of customer perceptions of quality management in the software industry: An empirical study in India, *Software Quality Journal*, 14(4), 291–308.

Järvinen, P., 2004. *On Research Methods*. Opinpajan Kirja.

Joan, R. & Pastor, A. J., 2007. "Applying Grounded Theory to Study the Implementation of an Inter-Organizational Information System", *Electronic Journal of Business Research Methods*, 5(2), 71-82.

Jones, M. & Alony, I., 2011. Guiding the use of Grounded Theory in Doctoral studies – an example from the Australian film industry. *International Journal of Doctoral Studies*, 6 (N/A), 95-114.

Jones, M. L., 2005. 'Lights… action… grounded theory': developing an understanding for the management of film production, *Rhizome*, 1 (1), 143-153.

Jones, M. L., Krilik, G. K. & Zanko, M., 2005. Grounded Theory: A Theoretical and Practical Application in the Australian Film Industry, In proceeding of the International Qualitative Research Convention.

Jørgensen, M., 1999. "Software Quality Measurement". *Advances in Engineering Software*, 30(2), 907-912.

Jung, J., Lee, S., Choi, S. & Lee, S-W, 2014. Requirements engineering process improvement: Analyzing the organizational culture impact and implementing an empirical study to evaluate the benefits of improvement, 2014 IEEE 1st International Workshop on the Interrelations between Requirements Engineering and Business Process Management (REBPM), Karlskrona, pp. 15-18.

Kalyani, M., 2013. *Setting Standards for the Murky Cloud Market*, https://spideroak.com/privacypost/business-the-cloud/cloud-computing-regulations-on-the-rise/. [Accessed on 7.5.2014]

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

Kitchenham, B. & Pfleeger, S. L., 1996. Software "quality: The elusive target". *IEEE Software*, 13(1), 12-21.

Kitchenham, B. & Pickard, L. M., 1987. Towards a constructive quality model. Part 2: Statistical techniques for modelling software quality in the ESPRIT REQUEST project. *Softw. Eng. J.* 2(4), 114–126.

Kläs, M., Heidrich, J.,Münch, J. & Trendowicz, A., 2009. CQML scheme: A classification scheme for comprehensive quality model landscapes. In: Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications.

Klein, H. K. & Myers, M. D., 1999. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, 23(1), 67-94.

Kusters, R. J., Van Solingen, R. & Trienekens, J. J. M., 1997. "User-perception of Embedded Software Quality", Proceedings of the STEP97 Conference, IEEE Computer Society.

Lagrosen, S., 2005. "Customer involvement in new product development: A relationship marketing perspective", *European Journal of Innovation Management*, 8(4), 424-436.

Lester, R., 2013. *Transitioning to Agile*, https://www.scrumalliance.org/community/articles/2013/march/transitioning-to-agile [Accessed on 26.11.2014].

Madabhavi, C., 2014. *Transitioning from Waterfall to Agile*, https://www.scrumalliance.org/community/articles/2014/july/transitioning-from-waterfall-to-agile, [Accessed on 26.11.2014].

Mahalingam, T. & Rajan, A. V. 2013. "Cloud and mobile computing: Affordances of the 21st century teaching and learning", International Conference on Current Trends in Information Technology (CTIT), Dubai, p.125-128.

Marcos, A. & Zagalo N., 2011. Instantiating the creation process in digital art for serious games design, Entertainment *Computing*, 2 (2011) 143–148.

Martin, P. Y. & Turner, B. A., 1986. Grounded theory and organizational research. *The Journal of Applied Behavioral Science*, 22 (2), 141-157.

Maxwell, J. A., 1992. Understanding and Validity in Qualitative Research, *Harvard Education Review*, Vol. 62, No. 3, pp. 279-300.

Mayer, C. B., 2001. A case in Case Study Methodology. *Field methods*, 13(4), pp. 329-52.

McCall, J. A., Richards, P. K. & Walters, G. F., 1977. *Factors in Software Quality*. National Technical Information Service, Springfield.

Miatidis, M., Jarke, M. & Weidenhaupt, K., 2008. Using developers' experience in cooperative design processes, Lecture Notes in Computer Science. 4970(2008), 185-223.

Moody, D., 2002. *Empirical Research Methods*, http://www.itu.dk/~oladjones/semester%203/advanced%20it%20mgt%20and%20software%20engineering/project/materials/what%20is%20empirical%20research1.pdf [Accessed on 5.11.2014].

Mudunuri, S. B, 2010. *Software Testing Methodologies Course Page*, http://www.mcr.org.in/sureshmudunuri/stm/unit1.php [Accessed on 29. 12. 2014].

Myers, M. & Klein, H., 2011. 'A Set of Principles for Conducting Critical Research in Information Systems', *MIS Quarterly*, 35(1), 47-58.

NASA, 2009. "*Software Assurance Definitions*", http://www.hq.nasa.gov/office/codeq/software/umbrella_defs.htm [Accessed on 11.10.2014].

New Oxford American Dictionary, 2 Ed., 2005. Oxford University Press, ISBN 0-19-517077-6.

Osterweil, L. J., 1997. Software processes are software too", The 19th International Conference on Software Engineering, Boston, pp. 343-344.

Pare, G. & Elam, J. J., 1997. Using case study research to build theories of IT implementation in information systems and qualitative research. Proceedings of The IFIP TC8 WG 8.2 Conference On Information Systems And Qualitative Research, Philadelphia, Pennsylvania, pp. 542-569.

Park, C., Pattipati, K. R., An, W. & Kleinman, D. L., 2012. "Quantifying the Impact of Information and Organizational Structures via Distributed Auction Algorithm: Point-to-Point Communication Structure, Systems, Man and Cybernetics, Part A: Systems and Humans", *IEEE Transactions*, 42(1), 68-86.

Paulk, M. C., 1995. *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley.

Paulk, M.C., Curtis, B., Chrissis, M.B. & Weber, C.V., 1993. Capability maturity model, version 1.1, *IEEE Software*, 10(4), 18–27.

Perner, L., 2008. http://www.consumerpsychologist.com/marketing_introduction.html, [Accessed on 17.10. 2014].

Petre, M., 2010. Mental imagery and software visualization in high-performance software development teams, *Journal of Visual Languages and Computing*. 21(2010), pp. 171–183.

Piffer, D., 2012. Can creativity be measured? An attempt to clarify the notion of creativity and general directions for future research. *Thinking Skills and Creativity*, 7(3), 258–264.

Robson, C., 2002. *Real World Research,* 2nd Ed. Oxford. Blackwell.

Salo, O. & Abrahamsson, P., 2008. Agile methods in European embedded software development organizations: a survey on the actual use and usefulness of Extreme Programming and Scrum, *Software, IET*, 2(1) 58 – 64.

Sangroha, D. & Gupta, V. Exploring Security Theory Approach in BYOD Environment, Wireless Networks and Security Proceedings of the Second International Conference on Advanced Computing, Networking and Informatics (ICACNI-2014), Kolkata, India, pp. 259-266, (2014).

Savolainen, J., Kauppinen, M. & Mannisto, 2007. T. Identifying Key Requirements for a New Product Line, (IEEE-APSEC 2007) 14th Asia-Pacific Software Engineering Conference, Aichi, pp. 478 – 485.

Scarfo, A., 2012. New Security Perspectives around BYOD, 7th International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA), Victoria, BC, pp. 446 – 451.

Schaefer, R., 2009. Software maturity: Design as dark Art. *SIGSOFT Software Engineering Notes*, 34(1), 1-36.

Seth, F. P., Mustonen-Ollila, E. & Taipale, O., 2014b. "The Influence of Management on Software Product Quality: An Empirical Study in Software Developing Companies", Proceedings of the 21st International Conference on European System, Software & Services Process Improvement & Innovation (EuroSPI), 25-27 June 2014, Luxembourg, Volume 425, 2014, pp. 147-158.

Seth, F. P., Mustonen-Ollila, E., Taipale, O. & Smolander, K., 2012. "Software Quality Construction: Empirical Study on the Role of Requirements, Stakeholders and Resources", Proceedings of the IEEE 19th Asia-Pacific Software Engineering Conference (APSEC), 4-7 December 2012, Hong Kong, pp. 17-26.

Seth, F. P., Taipale, O. & Smolander, K., 2014a. "Organizational and Customer related Challenges of Software Testing", Proceedings of the IEEE 8th International Conference on Research Challenges in Information Science (RCIS), 28-30 May 2014, Marrakesh, Morocco, pp. 1-12.

Shaw, M., 2002. What Makes Good Research in Software Engineering? International Journal of Software Tools for Technology Transfer, 4(1) 1-7.

Shneiderman, B., 2007. Creativity support tools accelerating discovery and innovation. *Communications of the ACM*, 50(12), 20 – 32.

Smolander, K., 2002. "Four metaphors of architecture in software organizations: finding out the meaning of architecture in practice". International symposium on empirical software engineering (ISESE 2002), Nara, Japan.

Strauss, A. L. & Corbin, J., 1990. *Basics of Qualitative Research: Grounded Theory Procedures and Applications,* Newbury Park, CA, Sage Publication.

Strauss, A. L. & Corbin, J., 1998. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques.* 2nd Edition. Sage.

Strecker, N., 2009. *Literature Review. Innovation Strategy and Firm Performance*, Gabler Verlag, GWV Fachverlage GmbH, Wiesbaden.

Suominen, M. & Mäkinen, T., 2013. On the Applicability of Capability Models for Small Software Organizations: Does the Use of Standard Processes lead to a Better Achievement of Business Goals? Software Quality Journal, DOI: 10.1007/s11219-013-9201-7.

Taipale, O. & Smolander, K., 2006. Improving Software Testing by Observing Practice, Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE), pp. 262-271.

Taipale, O., 2007. *Observations on software testing practice*. Doctoral Thesis, Acta Universitatis Lappeenrantaensis, Lappeenranta University of Technology Press.

Tayntor, C. B., 2003. "*Six Sigma Software Development*", CRC Press LLC, pp. 114.

Terzakis, J., 2013. The impact of requirements on software quality across three product generations, 2013 21st IEEE International Requirements Engineering Conference (RE), Rio de Janeiro, pp. 284 – 289.

Trivedi, P. & Sharma, A., 2013. A comparative study between iterative waterfall and incremental software development life cycle model for optimizing the resources using computer simulation, 2013 2nd International Conference on Information Management in the Knowledge Economy (IMKE), Chandigarh, pp. 188 – 194.

Tselentis, J., 2014. *Moving Beyond Waterfall to Agile Development + Design* http://www.howdesign.com/how-magazine/moving-beyond-waterfall-agile-development-design/#sthash.guiFG7zL.dpuf, [Accessed on 26.11.2014].

Van Manen, M., 1990. *Researching lived Experience: Human Science for an Action Sensitivity Pedagogy*. London: Althouse Press.

Wexelblat, R. L., ed., 1981. History of Programming Languages. New York: Academic Press. ISBN 0-12-745040-8.

Wohlin, C., Höst, M. & Henningsson, K., 2006. Empirical Research Methods in Web and Software Engineering, Web Engineering, Springer Berlin Heidelberg, pp. 409-430.

Wu, C & Simmons, D.B., 2000. Software Project Planning Associate (SPPA): a knowledge-based approach for dynamic software project planning and tracking, The 24th Annual International Computer Software and Applications Conference, COMPSAC 2000, Taipei, pp. 305 – 310.

Xenos, M. & Christodoulakis, D., 1997. Measuring perceived software quality, *Information and Software Technology*, 39(6), 417-424.

Yin, R., 2002. *Case Study Research: Design and Methods*, 3rd ed. Sage Publications.

Zeng, L., Proctor, R. W. & Salvendy, G., 2011. Can traditional divergent thinking tests be trusted in measuring and predicting real-world creativity? *Creativity Research Journal*, 23(1), 24–37.

Appendix I: Publications

ACTA UNIVERSITATIS LAPPEENRANTAENSIS

604. STOKLASA, JAN. Linguistic models for decision support. 2014. Diss.

605. VEPSÄLÄINEN, ARI. Heterogenous mass transfer in fluidized beds by computational fluid dynamics. 2014. Diss.

606. JUVONEN, PASI. Learning information technology business in a changing industry landscape. The case of introducing team entreneurship in renewing bachelor education in information technology in a university of applied sciences. 2014. Diss.

607. MÄKIMATTILA, MARTTI. Organizing for systemic innovations – research on knowledge, interaction and organizational interdependencies. 2014. Diss.

608. HÄMÄLÄINEN, KIMMO. Improving the usability of extruded wood-plastic composites by using modification technology. 2014. Diss.

609. PIRTTILÄ, MIIA. The cycle times of working capital: financial value chain analysis method. 2014. Diss.

610. SUIKKANEN, HEIKKI. Application and development of numerical methods for the modelling of innovative gas cooled fission reactors. 2014. Diss.

611. LI, MING. Stiffness based trajectory planning and feedforward based vibration suppression control of parallel robot machines. 2014. Diss.

612. KOKKONEN, KIRSI. From entrepreneurial opportunities to successful business networks – evidence from bioenergy. 2014. Diss.

613. MAIJANEN-KYLÄHEIKO, PÄIVI. Pursuit of change versus organizational inertia: a study on strategic renewal in the Finnish broadcasting company. 2014. Diss.

614. MBALAWATA, ISAMBI SAILON. Adaptive Markov chain Monte Carlo and Bayesian filtering for state space models. 2014. Diss.

615. UUSITALO, ANTTI. Working fluid selection and design of small-scale waste heat recovery systems based on organic rankine cycles. 2014. Diss.

616. METSO, SARI. A multimethod examination of contributors to successful on-the-job learning of vocational students. 2014. Diss.

617. SIITONEN, JANI. Advanced analysis and design methods for preparative chromatographic separation processes. 2014. Diss.

618. VIHAVAINEN, JUHANI. VVER-440 thermal hydraulics as computer code validation challenge. 2014. Diss.

619. AHONEN, PASI. Between memory and strategy: media discourse analysis of an industrial shutdown. 2014. Diss.

620. MWANGA, GASPER GODSON. Mathematical modeling and optimal control of malaria. 2014. Diss.

621. PELTOLA, PETTERI. Analysis and modelling of chemical looping combustion process with and without oxygen uncoupling. 2014. Diss.

622. NISKANEN, VILLE. Radio-frequency-based measurement methods for bearing current analysis in induction motors. 2014. Diss.

623. HYVÄRINEN, MARKO. Ultraviolet light protection and weathering properties of wood-polypropylene composites. 2014. Diss.

624. RANTANEN, NOORA. The family as a collective owner – identifying performance factors in listed companies. 2014. Diss.

625. VÄNSKÄ, MIKKO. Defining the keyhole modes – the effects on the molten pool behavior and the weld geometry in high power laser welding of stainless steels. 2014. Diss.

626. KORPELA, KARI. Value of information logistics integration in digital business ecosystem. 2014. Diss.

627. GRUDINSCHI, DANIELA. Strategic management of value networks: how to create value in cross-sector collaboration and partnerships. 2014. Diss.

628. SKLYAROVA, ANASTASIA. Hyperfine interactions in the new Fe-based superconducting structures and related magnetic phases. 2015. Diss.

629. SEMKEN, R. SCOTT. Lightweight, liquid-cooled, direct-drive generator for high–power wind turbines: motivation, concept, and performance. 2015. Diss.

630. LUOSTARINEN, LAURI. Novel virtual environment and real-time simulation based methods for improving life-cycle efficiency of non-road mobile machinery. 2015. Diss.

631. ERKKILÄ, ANNA-LEENA. Hygro-elasto-plastic behavior of planar orthotropic material. 2015. Diss.

632. KOLOSENI, DAVID. Differential evolution based classification with pool of distances and aggregation operators. 2015. Diss.

633. KARVONEN, VESA. Identification of characteristics for successful university-company partnership development. 2015. Diss.

634. KIVYIRO, PENDO. Foreign direct investment, clean development mechanism, and environmental management: a case of Sub-Saharan Africa. 2015. Diss.

635. SANKALA, ARTO. Modular double-cascade converter. 2015. Diss.

636. NIKOLAEVA, MARINA. Improving the fire retardancy of extruded/coextruded wood-plastic composites. 2015. Diss.

637. ABDEL WAHED, MAHMOUD. Geochemistry and water quality of Lake Qarun, Egypt. 2015. Diss.

638. PETROV, ILYA. Cost reduction of permanent magnet synchronous machines. 2015. Diss.

639. ZHANG, YUNFAN. Modification of photocatalyst with enhanced photocalytic activity for water treatment. 2015. Diss.

640. RATAVA, JUHO. Modelling cutting states in rough turning of 34CrNiMo6 steel. 2015. Diss.

641. MAYDANNIK, PHILIPP. Roll-to-roll atomic layer deposition process for flexible electronics applications. 2015. Diss.