DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS
**79**

DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS
**79**

**UUNO PUUS**

# Structural performance as a success factor in software development projects – Estonian experience

TARTU UNIVERSITY PRESS

Faculty of Mathematics and Computer Science, University of Tartu, Estonia

Dissertation is accepted for the commencement of the degree of Doctor of Philosophy (PhD) on April 23, 2012, by the Council of the Institute of Computer Science, University of Tartu.

Commencement will take place on June 11, 2012. at 09:15 in J. Liivi 2–404.

European Union
European Social Fund

Investing in your future

ISSN 1024–4212
ISBN 978–9949–19–999–0 (trükis)
ISBN 978–9949–32–000–4 (pdf)

Tartu Ülikooli Kirjastus
www.tyk.ee
Tellimus nr 234

# CONTENTS

# LIST OF ORIGINAL PUBLICATIONS

I. Heiberg, S., Puus, U., Salumaa, P., Seeba, A. (2003) Pair-Programming Effect on Developers Productivity, *Extreme Programming and Agile Processes in Software Engineering*, Lecture Notes in Computer Science, Vol. 2675, Springer, pp. 215–224

II. Puus, U., Seeba, A., Salumaa, P., Heiberg, S. (2004) Analyzing Pair-Programmer's Satisfaction with the Method, the Result, and the Partner, *Extreme Programming and Agile Processes in Software Engineering*, Lecture Notes in Computer Science, Vol. 3092, Springer, pp. 246–249

III. Puus, U., Mets, T., Torokoff, M., Tamm, A. (2009) Organizational Learning Environment in Software Industry – the Case of Estonian Enterprises, In *Proceedings of the 10th European Conference on Knowledge Management*, Vicenza, Italy, pp. 642–681

IV. Puus, U., Mets, T. (2010) Software development maturity evaluation: six cases from Estonian SMEs. *Baltic Journal of Management*, Vol. 5, No. 3, pp. 422–443

## PUBLICATIONS NOT INCLUDED IN THE THESIS

1. Puus, U., Mets, T. (2009) SECI Model and Software Life Cycle: the Organizational Learning View to Software Development Management, In *The ESU conference in Entrepreneurship 2009 "Many voices of European Entrepreneurship Research": Full papers, Riviezzo, A., Kyrö, P., Napolitano, M.-R. (Editors)* pp. 745–758
2. Puus, U., Mets, T., Torokoff, M., Tamm, A. (2009) Organizational Learning Environment in Software Industry – the Case of Estonian Enterprises. Abstract, In: *Proceedings of the 10th European Conference on Knowledge Management: The 10th European Conference on Knowledge Management*, Scarso, E., Bolisoni, E. (Editors), Vicenza (Italy), September 3–4, 2009, Academic Publishing Limited, p. 41
3. Puus, U., Mets, T. (2009) Organizational Learning View: Situation in Estonian ICT Enterprises, Abstract, In *Proceedings of Modern Management Research Conference (MMRC), Insights into the sustainable growth of business,* Vilnius; 19.–21.11.2009. Emerald Group Publishing Limited, Online http://www.ism.lt/mmrc/2009/articles_index.html, viewed 13.06.2011
4. Puus, U., Mets, T. (2009) SECI Model and Software Life Cycle: the Organizational Learning View to Software Development Management. Abstract. In *The ESU conference in Entrepreneurship 2009. Conference handbook: The ESU conference in Entrepreneurship 2009; Benevento, Italy; Sept. 8–13 2009,* Benevento: University of Sannio, p. 59
5. Palm, R., Peder, A., Kiho, J., Uibo, H., Jaeger, J., Puus, U., Salumaa, P., Meho, I., Tõnisson, E. (2003) *Programmeerimise praktikumid. Algklassid.* Tartu, Tartu Ülikooli Kirjastus
6. Heero, K., Puus, U., Willemson, J. (2002) XML based document management in Estonian legislative system. In *Databases and information systems: 5th International Baltic Conference BalticDB&IS,* Tallinn; 03.–06.06.2002. (Editors) Haav, H-M., Kalja, A. Tallinn, Tallinna Tehnikaülikool, 2002, 321–330
7. Oit, M., Puus, U. (2000) Andmebaaside turvalisus: tavamehhanismid. *A & A*, 6, 41–48
8. Puus, U., Pöial, J. Villems, A. (1987) *Arvutiõpetuse abimaterjal.* Tallinn: Eesti NSV Haridusministeerium

# ABSTRACT

Although the history of software development projects is long, the failure of projects remains a concern. One of the reasons for this situation is that the success of a software development project is defined by different stakeholders in different ways. A software development project usually has three main groups of stakeholders – the development company management, the project customer as software user and the project team as the creator of software. A successful project is completed, when software, which is user-friendly and works, is delivered on time and within budget. Hence, project success is defined in terms of a consensus between the interests of these three groups of project stakeholders.

To achieve the consensus mentioned above, projects developed different (management) structures during software development and applied different principles of management. One of these, agile methodologies, has attracted more attention in recent years. The agile approach values openness, frequent communication and the involvement of all stakeholders. Agility serves as a methodological basis for success in software development projects having a clearly defined structure of interaction (with the stakeholders) in the project. Such a structure makes it possible to achieve and establish a consensus of interests among the different stakeholders.

The current research focuses on structural performance as method to achieve the consensus between software project stakeholders. Structural performance incorporates elements of agile software development methodology including organizational structure, standards and other methodological choices established in the project. Such set of different policies is generalized as an instrument of the project management to improve process performance, and therefore, to complete software development projects successfully. Three instances of structural performance – 1) pair-programming, 2) organizational learning environment and 3) software development process maturity level based on self evaluation – are implemented to evaluate structural performance as a measure of agility. An analysis of structural and process performance among Estonian software development teams is carried out.

# 1. INTRODUCTION

The concept of the software development project is the same age as software development itself. The first Software Development (hereafter SD) projects were performed in the 1960s. Although the history of SD projects is long, the failure of projects remains a concern. According to Eveleens and Verhoef (2010) every fourth project is failing, however the percentage of failed projects is decreased from 31% in 1994 to 24% in 2009 (ibid). Since failure rates are usually relatively high, then "no failure" can hardly be considered a success criterion. In other words, every more or less properly completed SD project could be considered successful. One of the reasons for this situation is that the success of an SD project is defined by different stakeholders in different ways. The most common groups of stakeholders are (Rising and Janoff 2000; Hoffman and Lehner 2001): 1) software customers, 2) the SD project team, and 3) the SD company management. Software customers need an information system, the project team is interested in inspiring work and the company management is responsible for profitability. Therefore, SD process is relatively controversial. The SD discrepancy triangle is presented in Figure 1.1.

The discrepancy is concealed in the fact that it is actually impossible perfectly to fulfil the time, cost and functionality and usability targets in SD simultaneously. When the necessary functionality is delivered and usable on time, this very often results in exceeding the budget, the agreed functionality within budget results in late delivery and a project that is within budget and on time is only possible at best with limited functionality (sometimes delivering a useless system). Every SD project is faced with these problems, especially when deadlines are approaching.

A successful SD project results from an effective solution to this discrepancy. There are different methodologies and approaches describing more or less suitable solutions for the success of SD projects. Regarding development models, historically the first was the waterfall model (Royce 1987), and the latest was the agile approach since the beginning of the last decade (Cockburn 2002). The agile approach characterizes *inter alia* team development and organizational learning in SD teams. The common elements and features of agile development and organizational learning are presented in (Qumer and Sellers 2010; Salo and Pikkarainen 2005; Pikkarainen, Salo and Still 2005 and Salo 2007). Although agile methodologies will have been in use for around 10 years, the common, standardized methodology to use them in software development is still missing. In the sense of standardization, the best model is still the Capability Maturity Model and its subsequent development, the Capability Maturity Model Integrated (hereafter CMMI) (CMMI 2006). The use of CMMI as an SD model in the deployment of agile practices is described in (Kähkönen and Abrahamsson 2004; Turner 2002; Clazer, Dalton, Anderson, Konrad and Shrum 2008 and Kalermo and Rissanen 2002).

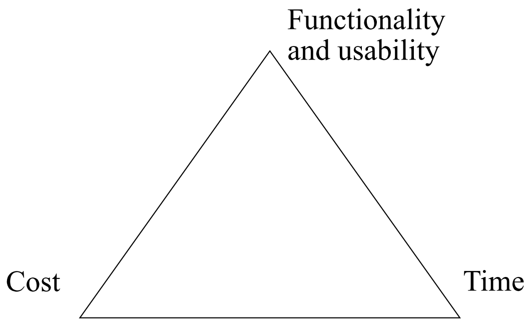


**Figure 1.1** SD discrepancy triangle

Source: Composed by the author

SD project success is also interpreted in terms of conflict management (Gobeli, Koenig and Bechinger 1998), the mathematical causal model (Procaccino, Verner, Darter and Amadio 2005) or as a project manager leadership issue (Garcia-Morales, Llorens-Montes and Verdu-Jover 2008; Kivipõld and Ahonen 2011).

With a view to assess the quality of SD, the process is created in accordance with standards, for example, using the CMMI. The SD team as a Learning Organization (LO) is described in (Fichman and Kemerer 1997; Kelly 2003; Jiang, Klein, Hwang, Huang and Hung 2004 and Shepherd, Tesch and Hsu 2006).

Project success is related also to another aspect – project performance. As in project success, project performance has the same dimensions as outlined for project success above – a management dimension, a customer dimension and a project team dimension. However, project success and project performance are measured differently. The concept of *success* is used hereafter in the sense of perceived success that stakeholders have perceived based on the outcome of the project. This success is a rather emotional measure and is explained regardless of deadlines and project cost via the feelings and emotions of the project participants. Project performance is explained in a more quantitative way using terms like deadlines (as Cycle Time), costs (as Effort), size of system (as Lines of Code) and quality of development (as Errors Repaired) (Harter, Krishnan and Slaughter 2000), and therefore, are more suitable for analysis.

Project performance is divided into structural and process components. Structural and process performances were firstly differentiated by Nidomulu and Subramani (2003). Process performance is the quantitative part of performance, and is usually measured as project performance in general. Structural performance by contrast is not so well known and describes organizational structure and standards and other methodological choices established in the project.

Goal of this thesis is to explore the role of structural performance in the SD. For this purpose three different studies are performed. The problem statement, research process and structure of the thesis are presented in the following sections.

## Problem statement

Project success is based on several mutually interacting success factors. Performance as a factor of project success is an implemented capability carried into effect. According to Feldt, Angelis, Torkar and Samuelsson (2010) the capability of an SD team to perform a software project successfully is divided into 1) *capacity* as the physiological and cognitive abilities of the individual that enables him/her to perform a work task in an efficient way, 2) *willingness* as the psychological and emotional characteristics that influence the degree to which the individual is inclined to perform the task, and 3) *opportunity* as the particular configuration of the environment and surroundings beyond his/her direct control, an individual and his/her task that enables or constrains his/her performance. It is important to know which *features help different capabilities within SD teams actually to function, and so determine the success of the SD project*.

In above-mentioned terms of performance, capacity, willingness and opportunity set up the structural performance framework within the SD project. Consequently, project success is determined by structural performance as features carrying project capabilities into effect via process and project performance as presented in Figure 1.2

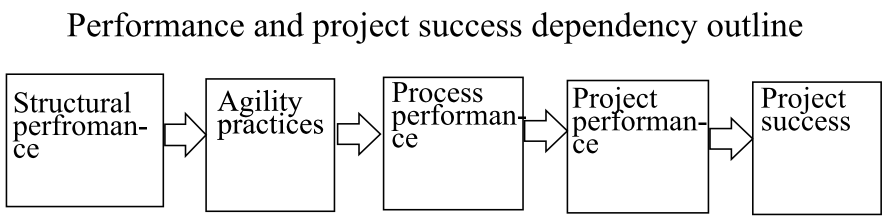


**Figure 1.2.** SD project performance and success

Source: Composed by the author

SD performance in general has structural and dynamic components. In mutual interaction, structural performance is coherently interrelated through agility practices with process performance as the basis for general performance. The latter, in turn, serves as a factor of the success of the SD project.

Agile practices as process management and control principles constitute an important part of structural performance. Agile practices if used in project management increase the flexibility of the project, and therefore, improve the adaptability of the project team in terms of structural performance. In other words, the structural performance of the SD team (as an organization) is measured according to their ability to adapt to changes in the internal and external environment.

The success rate of SD projects is an important competitive advantage for any SD company. Although the Estonian SD industry has developed substantially during the last decade, the current situation in the Estonian SD sector is complicated. Since the last years of the previous century there have been various challenges such as setting up complicated government information systems, creating banking information systems for new banks and creating world famous IT trademarks such as Skype and Playtech. First of all, it is necessary to analyse the experiences collected during these development years aiming to increase competitiveness in foreign markets and exports of SD products. In particular, there is a need for the analysis of present-day methodologies based on agile principles in Estonian SD teams. Therefore, dynamic Estonian SD teams need new approaches to analyze SD project performance.

*The goal of this thesis is to explore, describe and evaluate the structural performance as an essential component of project performance, and therefore, a project success factor in general on the example of Estonian SD teams.*

To achieve this goal the following research tasks were formulated:
1. Analysis and evaluation of SD project performance as a factor of the success of SD projects.
2. Analysing practices and creating suitable methods for evaluating structural performance in SD projects.
3. Clarifying the interaction between the structural and process components of SD performance in Estonian SD teams.
4. Exploring and understanding organizational patterns in Estonian SD teams in terms of agile practices, and at the same time in terms of structural performance.
5. Evaluating the maturity level of Estonian SD teams as a quality of structural performance.

SD project performance and the interaction between structural and process performance are analysed in the second chapter and also in *Study I*. The Organizational Learning (hereafter OL) patterns in SD teams are explored in *Study II* and the maturity level evaluation methodology is developed in *Study III*.

## The research process and methods

The thesis contains three relatively independent, but conceptually bound studies. The general goal of the research was to explore the interaction between structural and process performance in a more detailed manner. The research process was started in *Study I* at the initial (cellular) level, where the project team consisted of only two people. The goal of this study was to describe the interaction between structural (pair, non-pair) and process (productivity) performance in a minimal structure. In this study, it was intended to evaluate the productivity according to two different team structures – pair programming and non-pair programming. The research method was a controlled experiment.

In the same study (*Study I*), the minimal environment of the SD team was also described via the satisfaction level of the participants in the experiment. To describe the level of satisfaction, an appropriate questionnaire was used as the research method. As a result, three types of satisfaction were differentiated as perceptions of the development environment by the team members.

To clarify the linkages of structural performance and the SD project environment in a non-experimental (everyday SD) situation, a more advanced OL paradigm was chosen in *Study II* to explore the actual SD environment. The research method was a questionnaire followed by factor analysis to describe the situation in Estonian SD projects. Since the SD team is usually a part of a company, aspects of the enterprise were studied in *Study II* as well.

For practical implementations and in order to have a more precise image of the structural performance, it is necessary to evaluate the level of structural performance. A qualitative method for evaluating structural performance is described in *Study III*. The necessary case studies (semi-structured interviews) were performed in six Estonian Small and Medium sized Enterprises (SMEs). *Study III* demonstrated that evaluating the level of maturity in SD using self-assessment based on the CMMI process area categories, is a relevant method for evaluating structural performance in an SD project.

The schedule of the research process is represented in Figure 1.3 as sequential research steps and appropriate research methods to clarify and describe the structural performance outline.

The research process is initiated to describe the SD process at the initial (team consisting of two people) level. *Study I* describes the interaction of structural and process performance and the SD environment in terms of team member satisfaction. Since satisfaction is only one particular aspect, it was reasonable to introduce the description of a more complete SD environment. In *Study II* the respective environment was described as the Organizational Learning Environment (OLE). *Study II* complements the controlled experiment (*Study I*) with research of the environment of a real SD process. *Study II* is an extension of *Study I* in a real SD environment. In this study the environment of SD was described in terms of the structural performance of the organization (SD team). An exploratory approach was used to simultaneously describe the environment and the interaction of process and structural performance.

*Study III* aimed to evaluate the level of structural performance in the project team in order to have a more precise view of the quality of the specific working environment. The evaluation method was derived from CMMI and the level of structural performance was evaluated in terms of SD maturity. Project managers participating in *Study III* performed a self-evaluation of the maturity of the development process across CMMI process areas.

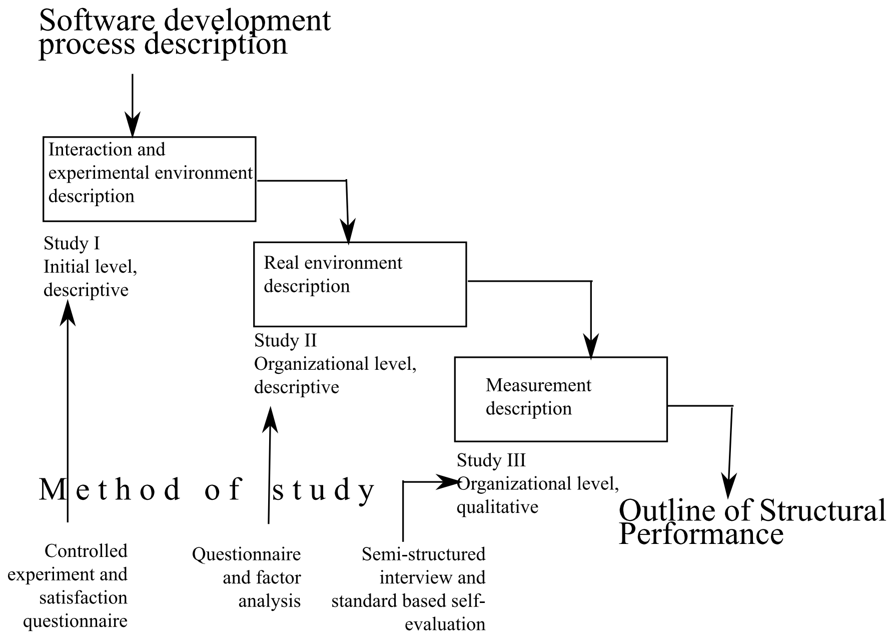

Software development process description
Interaction and experimental environment description
Study I Initial level, descriptive
Real environment description
Study II Organizational level, descriptive
Measurement description
Study III Organizational level, qualitative
Outline of Structural Performance
Method of study
Controlled experiment and satisfaction questionnaire
Questionnaire and factor analysis
Semi-structured interview and standard based self-evaluation

**Figure 1.3.** The research process

Source: Composed by author

As the result of the exploratory research process, the structural performance of the SD team was described as a five- and three-factor model of OLE, and qualitatively evaluated in terms of development maturity.

## Structure of the thesis

This study is based on four papers published between 2003 and 2010.

1. Heiberg, S., Puus, U., Salumaa, P., Seeba, A. (2003) Pair-Programming Effect on Developers Productivity, *Extreme Programming and Agile Processes in Software Engineering*, Lecture Notes in Computer Science, Vol. 2675, Springer, pp. 215–224

The author's main contribution was the description and analysis of the role of the personality of team members in pair-programming. In addition, the author also contributes to the methodology of the study and the experiment set up.

2. Puus, U., Seeba, A., Salumaa, P., Heiberg, S. (2004) Analyzing Pair-Programmer's Satisfaction with the Method, the Result, and the Partner, *Extreme Programming and Agile Processes in Software Engineering*, Lecture Notes in Computer Science, Vol. 3092, Springer, pp. 246–249

The paper is mostly the author's own work. The author contributed the rationale for the theoretical background, methodology, questionnaire development and data analysis. Personality data was collected during the experiment performed in cooperation with the co-authors. The co-authors also contributed to the preparation of the text.

3. Puus, U., Mets, T., Torokoff, M., Tamm, A. (2009) Organizational Learning Environment in Software Industry – the Case of Estonian Enterprises, In *Proceedings of the 10th European Conference on Knowledge Management*, Vicenza, Italy, pp. 642–681

The author contributed to the theoretical background and partially also the methodology and data analysis. The main part of the questionnaire was prepared at the Centre for Entrepreneurship at the University of Tartu and developed by author. Data collection, theoretical rationale and part of the data analysis were carried out together with the co-authors. The text was prepared in cooperation with the supervisor.

4. Puus, U., Mets, T. (2010) Software development maturity evaluation: six cases from Estonian SMEs. *Baltic Journal of Management*, Vol. 5, No. 3, pp. 422–443

The paper is mostly the author's own work. The theoretical framework, methodology, interviews and data analysis was the author's contribution. The text was prepared in cooperation with the co-author.

Copies of the papers are included in the thesis on pages 49–100. The four papers included in the thesis are divided between three studies. *Study I* is based on papers 1 and 2, *Study II* on paper 3 and *Study III* on paper 4.

The outline of the thesis is as follows. The first chapter is the introduction. The main body of the thesis consists of six chapters. The second chapter includes different definitions of success in software development projects and concept of agility as a pattern of interaction between different types of performance. The concepts of software development success are generalized and a definition of SD development success is presented based on the consensus of SD project stakeholders. The relationship between organizational performance and process performance is clarified, and structural performance improvement is introduced as a managerial tool for process performance enhancement.

The third chapter defines and clarifies structural performance as a success factor of SD projects. Different instances of structural performance are introduced and a structural performance evaluation methodology is described based on SD development standards. The fourth chapter is dedicated to presenting and discussing the findings explored in the studies. Conclusions are formulated in the chapter five and limitations of the study in the chapter six.

# 2. SOFTWARE DEVELOPMENT PROJECT SUCCESS IN AN AGILE ENVIRONMENT

## The traditional vs the agile approach

In the current thesis, the traditional approach is interpreted as the methodologies and models used before the publication of the Agile Manifesto (Fowler and Highsmith 2001). Most of these essential software development processes are presented in CMMI. In this sense CMMI is a software development model based on best practices in traditional software development. Additionally, CMMI contains definitions of different areas of SD including the Software Improvement Process. Hence the SD model presented by CMMI is based on best (historical) practices in traditional software development.

Historically, the agile approach was already introduced in 1995, in the Quality Improvement Paradigm proposed by Basili and Caldera (1995). Basili and Caldera presented two closely interacting iterative learning cycles 1) the SD project as a learning organization, and 2) the whole company as a learning organization. Different authors (Kähkonen and Abrahamsson 2004; Turner 2002; Glazer, Dalton, Anderson, Konrad and Shrum 2008) compared and analyzed traditional (CMMI based) and agile approaches.

SD project success criteria differ for different stakeholders. For example, a project is successful for the company management when the project is finished on time, with the agreed costs and the delivered software implements all the agreed features in a manner that satisfies the customer (Procaccino, Verner, Overmyer and Darter 2002). The SD team involved in the project has a different attitude on some occasions. Usually, developers also appraise the information system created as a result of the project, technological challenges surmounted during the project and skills or experiences obtained. Likewise, the satisfaction of the development team is important. Organizational performance aspects related to human resources practices are described by Tseng and Lee (2009).

The project team view is explained, for example, in the survey by Procaccino, Verner, Shelfer and Gefen (2005). According to the results of this survey, successful SD projects have: 1) customers/users that provide feedback, 2) sufficiently skilled development team, and 3) feedback provided by the project manager. The fact that a successful project implies the following characteristics: 1) the developers have a sense of achievement, 2) they have a good job, 3) this job results in professional growth, and 4) they have learned something new, was also mentioned (ibid.).

In addition to the company management vs development team dimension project success determinants[*] are divided into personal (internal) and organizational (external). Personal determinants are 1) competency, 2) personal

[*] Hereafter in current study we use concepts *success factor* and *success determinant* as synonyms.

characteristics, 3) communication and negotiation, 4) societal culture, and 5) learning and training. Organizational are external determinants, such as 1) customer satisfaction, 2) customer collaboration, 3) customer commitment, 4) decision time, 5) team distribution, 6) team size, 7) corporate culture, 8) planning, and 9) control (Misra, Kumar and Kumar 2009).

Project success is reciprocally related to another concept – project performance. For the company management project success and project performance are usually synonyms – high project performance is in the opinion of the managers a prerequisite for project success. Although the company management and the project team should have the same objectives, in reality they sometimes differ. In the SD process some kind of discrepancy between the development team and the company management is observed. This discrepancy is described by one developer who said in an interview, "The rules of the organization favor revenue generators, and not instruments. I like working with instruments, especially embedded software" (Linberg 1999: 188). Indeed the goal of the company is to have proper revenue and earn a profit, and the interests of the developers and the firm need to be considered.

Another list of project success determinants is presented by Procaccino et al (2002) as follows: 1) management, 2) customers and users, 3) requirements, 4) estimation and scheduling, 5) the project manager, 6) the SD process, and 7) development personnel. The difference between the views of the management and the development team is stressed by Procaccino et al (ibid.) as well. It is essential for the management to deliver a software product in accordance with the business goals, in time and within budget. By contrast, the project team evaluates the clarity – for the development team the project should be completed (despite the necessary expenses) or cancelled (despite the business results of cancelling).

Cancelling projects reduces the profitability of the company, but from the development team's perspective a cancelled project is a source of experience. A comparison of completed and cancelled projects in the project success continuum is presented in Table 2.1

The software developers' view demonstrates the benefits of cancelled projects as sources of experience, that for the management are sources of loss. Similarly, the management's view of project success is not coherent. The measurable indicators such as deadlines and budget can easily be measured, but correspondence to business goals is more complicated to evaluate. Business goals depend not only on the development team or the company management, but also on a third party – the customer or software product user. Although one of most important business goals is customer satisfaction, software is some kind of special product in the sense of customer satisfaction. The reason for this is that it is usually too complicated for the software customer to change their information system. So sometimes the first versions of software are convenient and usable for customer, but after some period the software provider is unable to maintain the delivered software. In this case the customer has to make difficult choices about whether to continue with a problematic system or switch to a new one, taking into account that introducing the new system needs new investment.

**Table 2.1** Comparison of cancelled vs completed project success determinants

| **Project outcome** | **Failure** | **Low success** | **Successful** | **High success** | **Exceptio-nally successful** |
|---|---|---|---|---|---|
| Project completed | Developing a product that causes customer discontent (not meeting quality expectations) | Below average cost, effort and schedule performance compared to industry AND meeting quality expectations | Average cost, effort and schedule performance compared to industry AND meeting quality expectations | Better than average cost, effort and schedule performance compared to industry AND meeting quality expectations | Meeting all quality, cost, effort and schedule expectations |
| Project cancelled | Not learning anything that can be applied to the next project | Learning can be minimally applied to future projects | Learning can be applied to future projects. Some elements from the cancelled project can be directly used on future projects | Substantial learning can be applied to future projects. Significant numbers of elements from the cancelled project can be directly used on a future project | A cancelled project cannot be called "exceptio-nally successful" |

Source: Linberg, K. R. (1999)

Customer relationships add a new dimension to the software success frame-work. The customer expectations and software developer relationships in the context of SD project success are described for example by Petter (2008).

Consequently SD project success is represented as the consensus between three different parties – the company management, the development team and the software customer as seen in Figure 2.1

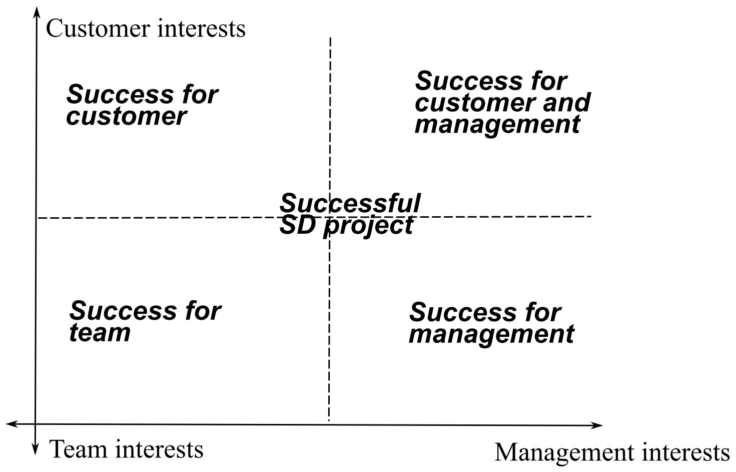


**Figure 2.1** Successful SD project as a consensus between management, team and customer

Source: Composed by the author

The project success for the management team includes reaching business goals, product delivery promptness and controlled costs. The project team on the other hand values challenges and the satisfaction of overcoming them and the knowledge and experience gained. For the software customers the situation is complicated and varies widely. For example, the customers are anxious to have the information systems they need, at the lowest price. Development service providers on the other hand are trying to compensate their investments in technology or the development process with the customer's money.

Those goals (listed above) should be achieved during the development process by creating the necessary information system and building mutual understanding simultaneously. As result, a successful SD project can be defined as a consensus between all stakeholders.

## Project performance as part of organizational performance

Another concept related to project success is project performance, usually defined (Jiang et al 2004) as SD team performance. As with project success, project performance has a different meaning for different stakeholders. Project performance is evaluated by the development team using several determinants as follows: 1) work quality, 2) team operations, 3) ability to meet project goals, 4) extent to which design objectives can be met, and 5) the reputation of the work excellence (Faraj and Sproull 2000). The company management by comparison values: 1) project costs and 2) time to completion (ibid). Project success factors are divided into internal and external determinants (Tomaszewski and Lundberg 2005). The relationship between project success and project performance is presented in Table 2.2

Concerning SD success, it is mentioned "... practitioners consider software projects successful if they provide intrinsic, internally motivating work to develop software systems that both meet customer/user needs and are easy to use" (Procaccino et al 2005: 200). In the case of project performance, it is essential that the software be acceptable for the customer, developed in time and within the planned budget. Consequently, a successful SD project has good project performance according to the management and the development team.

**Table 2.2** Relationship between project success and performance

| | Project success determinants: Procaccino et al (2005) | Project success determinants: Misra et al (2009) | Project performance determinants: Faraj and Sproull (2000) |
|---|---|---|---|
| Management view, external or organizational determinants | 1) customers that provide feedback, 2) sufficiently skilled development team and 3) feedback provided by project manager | 1) customer satisfaction, 2) customer collaboration, 3) customer commitment, 4) decision time, 5) team distribution, 6) team size, 7) corporate culture, 8) planning 9) control | 1) project costs and 2) time to completion |
| Development team view, internal or personal determinants | 1) the developers have a sense of achievement, 2) they have a good job, 3) this job results in professional growth and 4) they have learned something new | 1) competency, 2) personal characteristics, 3) communication and negotiation, 4) societal culture, 5) learning and training | 1) work quality, 2) team operations, 3) ability to meet project goals, 4) extent of meeting design objectives, 5) reputation of work excellence |

Source: Composed by the author, based on (Procaccino et al 2005; Misra et al 2009; Faraj and Sproull 2000)

Project success and performance are measured differently. Success is a more emotional measure and is explained regardless of the deadlines and project cost via the feelings and emotions of the project participants. Project performance is explained in a more quantitative manner. For example, in addition to deadlines (as Cycle Time) and costs (as Effort), project performance is often measured according to Lines of Code or Errors Repaired (Harter et al 2000). Project performance, as with project success, has the same dimensions as described above – a management dimension, a customer dimension and a project team dimension.

The customer and management dimension are primarily extended to different levels of management such as the operational, tactical and strategic levels. The current study focuses on the operational level because the management of a specific software development project is usually an example of operational management.

Project performance and organizational performance are also essential abilities of the organization (SD company) for performing SD projects successfully. In the literature, organizational performance as a measure of the quality of the organization is described in several ways.

(1) At the level of the organization or firm, and in the sense of business processes, organizational performance is described as turnover, productivity and corporate financial performance (Huselid 1995). *These aspects are essential for the management of an SD company and project.*

(2) Additionally, organizational performance is explained by market performance (Delaney and Huselid 1996), (Lai and Cheng 2005). *Market performance (as customer feedback and involvement) and productivity performance (as usability and viability of the delivered product) are related to customer success in the sense of SD project success.*

(3) Employee performance, employee innovation and employment relations are important at the organizational level as well (Guest, Conway and Dewe 2004). *So employee performance, employee innovation and employment relations in organizations are related to the SD team at the SD project level.*

According to statements (1), (2) and (3) above, there exists clear links between project performance dimensions (management dimension, customer involvement dimension, project team dimension) and organizational performance dimensions. Therefore organizational performance measures describe different aspects of business processes similarly to project performance measures.

Despite the clear connection between organizational performance and project performance, these concepts are different. SD project performance is narrower and project specific – it has sense mostly in the SD project context.

The performance of an SD project is divided into structural and process components. Structural and process performance were firstly differentiated by Nidomulu and Subramani (2003), where the structural approach is explained as the structure of the project team "... describing organizational structuring – the pre-specification of standards and choices regarding the level of delegation of decision-making". In contrast, the process based approach is explained as "…

specification of behaviours and outcomes as the key means to guide work" Nidomulu and Subramani (ibid).

The management and customer dimensions are important as the wider environment, but while structural performance, as well as process performance, is defined in the current study for the project team, the focus is the team level. The relationship between different types of performance is presented in Figure 2.2

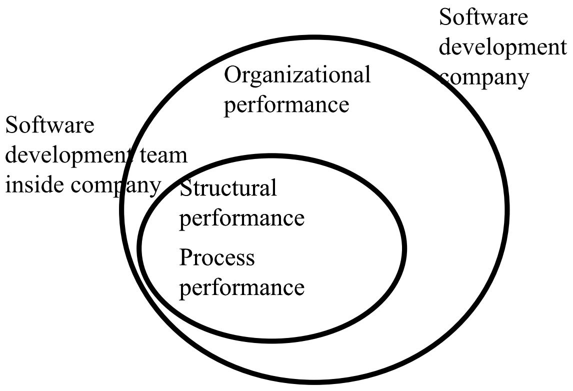

Software development company
Organizational performance
Software development team inside company
Structural performance
Process performance

**Figure 2.2** The relationships between types of performance

Source: Composed by the author

Organizational performance is more general and characterizes the organization (company, firm) and business process as a whole. Consequently, the SD company as a whole is characterized by organizational performance. The SD team as part of the organization (inside company) is usually project specific, and therefore, more restricted. Since SD is part of the business process, development performance forms a corresponding part of organizational performance. SD performance is, in turn, divided into structural and process performance.

## Agility as a pattern of interaction between different types of performance

According to Nidomolu and Subramani (2003), the structural and process approach explain two different but mutually interacting aspects of project performance. The question is how these two approaches interact in an actual SD process in an SD team? Evidence of a positive correlation between the structural and process component of performance is described in Harter et al (2000), where characteristics of process performance, such as the number of

Errors Repaired, Lines of Code (delivered in a certain period) are positively correlated by evaluations of structural performance like maturity level using the Capability Maturity Model Integrated (CMMI).

The SD process evaluation method developed in the current study describes five levels of maturity in the SD process expressed through a set of key process areas that should be in place for each level. The OL framework is described as key features, based on process areas of the environment where the OL appears. The OL framework is applicable in the SD team because SD itself is a knowledge intensive activity and needs to adapt new technologies and improve practices (Mathiassen and Pourkomeylian 2003).

OL is usually defined as a cyclical process (Argyris 1976). In the sense of the SD process, a repeating cycle proceeds from new information from the development environment and feedback about the present performance of the development process, which creates new knowledge. This may lead to the formation of new (infra)structure – developing structural performance. Created concepts sometimes lead to the creation of new learning strategies and infrastructure as new levels of structural performance.

The new wave of methodologies in software development called the agility approach has been in use since the beginning of the 00s. The management of SD team agility is defined by Qumer and Sellers (2010: 504) as "... a persistent behaviour or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment."

Definitions of agility (given above) and those for organizational learning mostly overlap. Lets focus here on the definition of organizational learning proposed by Argyris and Schön (1978: 18) "… learning is characterized as when, members of the organization respond to changes in the internal and external environment of the organization by detecting errors which they then correct so as to maintain the central features of theory-in-use". Single-loop learning is also defined by Argyris and Schön and they differentiate second-loop learning as "… those sorts of organizational inquiry which resolve incompatible organizational norms by setting new priorities and weightings of norms, or by restructuring the norms themselves together with associated strategies and assumptions" (ibid: 18). So single-loop learning means the adaptation of organizational members to suit the internal and external environment. At the same time, second-loop learning relates to the development of the environment according to internal and external changes.

Both, agility and organizational learning are oriented to adapting to a changing environment. Organizational learning by definition has no limitations about conditions and methods, or about how learning should occur. In contrast, agility is defined as a simple adaptation with minimum time and resources.

Ibert (2004) explains organizational learning in project-based organizations as being divided into linear and cyclical time concepts. The SD team uses the linear time concept because of the time limitation of the project. As a result, quite a large proportion of the experience gathered during one concrete project is not usable in the same project, but is applicable in the following project or projects. Then the linear concept of time could also only describe the linear (without feedback loops) organizational non-learning process.

Knowledge, generated in a particular project, can be transferred to subsequent projects, and therefore knowledge and experience, collected during one project, are reasonable to save for future use to ensure development and learning in the SD team. Frameworks, employed to manage knowledge, are explored by theory of OL. Most known of these frameworks are five disciplines introduced by Senge (1990). Senge differentiates five main features of OL as follows (ibid):

1. *Systems thinking* – it is the feature that integrates the others OL features, fusing them into a coherent body of theory and practice ,
2. *Mental models* are deeply ingrained assumptions, generalizations, or even pictures and images that influence how team members understand the project and how they take action,
3. *Personal mastery* – it is a lifelong discipline. People with a high level of personal mastery are acutely aware of their ignorance, their incompetence and their growth areas,
4. *Team learning* – Team learning starts with "dialogue", the capacity of members of a team to suspend assumptions and enter into a genuine "thinking together",
5. *Shared vision* – is "pictures of the future" that foster genuine commitment and enrollment rather than compliance.

Structural performance and at the same time OL features, first of all *Systems thinking* and *Mental models,* allow the storing of knowledge and experience in a retrievable way. Hence structural performance links the linear concept of time in a project with the cyclical concept of time in a company or firm. While only the cyclical concept of time is applicable for describing organizational learning, structural performance implements knowledge transfer from the project level to the company level as OL.

In addition, the fact that cyclical concepts appear quite often in projects should also be mentioned – iteration within a project is a typical example of a cycle within a project.

Senge's OL features are as presumptions for application agility in organizations including SD teams. Different authors are stressing the importance of *Mental models* (Kruchten 2007; Kollmann 2008; Cao and Ramesh 2007) and *Shared vision* (Kollmann 2008).

According to several authors (Rifkin and Fulop 1997; Johnson 2002; Mitki, Shani and Meiri 1997), OL is 1) an OL *process*, including activities to structure, save and reuse the knowledge created during a project, taking place in 2) a learning organization; that is, an OL *environment* that provides the rules, structures and technologies to manage this knowledge. In terms of SD performance, process performance characterizes the OL process and structural performance characterizes the (agile) OL environment accordingly. In organizations and also in SD team Learning Organization forms the main part of environment for OL.

Single-loop and double-loop learning are differentiated according to the action with different governing variables. In the SD project, the governing variables in single-loop learning are 1) project plan, 2) cost and 3) deadlines (Cao and Ramesh 2007). Since the SD project usually takes place in a rapidly changing environment, the simultaneous achievement of all governing goals/ variables is impossible. Therefore, single-loop learning has some additional (implicit) governing variables like 1) maximizing gains and minimizing losses, 2) minimizing negative emotions, 3) rationality (ibid.), to ensure who among the stakeholders are gaining and who losing.

Argyris (1976) describes single-loop learning from the viewpoint of the organization member as follows: "The primary strategies are to control the relevant environment and tasks unilaterally and to protect themselves and their group unilaterally" (ibid: 368). This leads to "Control as behavioural strategy influences the leader, others, and the environment in that it tends to produce defensiveness and closeness, because unilateral control does not tend to produce valid feedback. […] Under these conditions, problem solving about technical or personal issues is rather ineffective" (ibid). Therefore, single-loop learning for SD is inappropriate or, in other words, single-loop learning is one of the reasons for the high percentage of unsuccessful SD projects.

The problems related to single-loop learning are possible to avoid by changing governing variables. In agile development and second-loop learning the governing variables are 1) valid information, 2) free and informed choice, and 3) internal commitment (Cao and Ramesh 2007; Argyris 1976). The suitability of the management model based on double-loop learning is stressed by Argyris (1976): "The behavioral strategies involve sharing power with anyone who has competence and with anyone who is relevant to deciding or implementing the action, in the definition of the task or the control over the environment" (ibid: 369).

The interdependency between OL as a general framework and learning loop levels is depicted in Figure 2.3

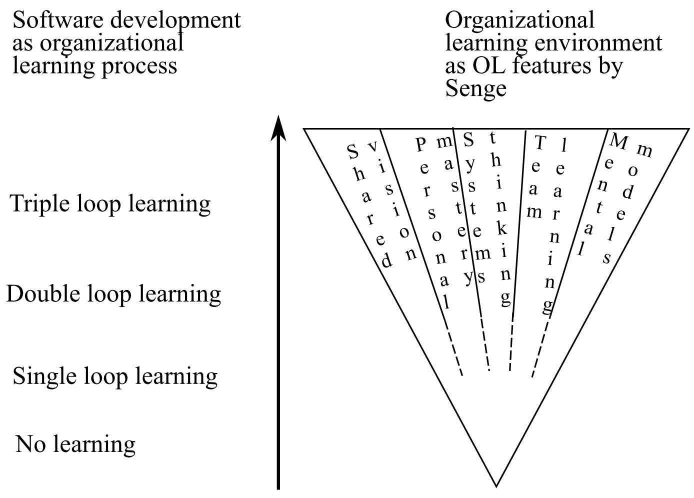


**Figure 2.3.** Organizational learning and SD process interdependency

Source: Composed by the author

On the left side (Fig. 2.3), the different levels of the learning process are presented from non-learning to triple-loop learning – the arrow shows the direction of the growth of the distinctness of LO patterns. On the right side, a general OL environment is described using the features of an LO introduced by Senge (1990).

At the *Initial* level no learning occurs. The project is managed in a chaotic way, the project team is not able to take corrective actions in reply to change or repeat their successes. OL features are not developed – they are at the embryonic stage.

*Single-loop* learning occurs when the project is *Managed*. The project is planned and executed in accordance with policy. Possible mismatches with policy are discovered and corrective actions are performed. According to the requirements of managed software development, the features of LO *Systems thinking* and *Mental models* exist, and the *Personal mastery* feature among project team members is developed.

In the case of *Double-loop* learning, the standard activities in a project are *Defined*. The standards, procedures, tools and methods used in the project are described and tailored. *Systems thinking* and *Mental models* are clear enough to create and use the necessary standards. *Personal Mastery* in connection with *Team learning* is developed to a suitable level to ensure the use and tailoring of the necessary standards. *The shared vision* among the project team is accepted by the team members as a common guide for performing the project.

In triple loop learning level Continuous *Optimizing* takes place in the project. A clear set of OL features is distinguishable. For example, the *Shared* *vision* is explained in qualitative-quantitative terms; *Personal mastery* and *Team learning* progress is expressed using qualitative-quantitative measures.

According to the framework described above, and in the sense of project performance divided into process and structural performance, the OL environment in the SD project is defined as the structural performance of this project. Consequently, any improvement of the OL environment means raising structural performance.

The improvement of the OL environment requires effort. In a project at the *Initial* level of maturity, whole such effort goes directly into the project outcomes; no resources are available to develop structural performance. To achieve the higher levels of structural performance it is necessary to create a suitable environment for OL and invest some resources (for example time) in this. Greater structural performance needs more resources to create the necessary environment. Hence, optimal structural performance comes from investing the resources to create an optimal environment to proceed with the project at the necessary or agreed (with the customer) level of maturity.

The mutual interaction between structural and process performance is described in Figure 2.4. Structural performance explained, for example, in terms of Process Maturity, Personal Satisfaction, Organizational Learning and so on, is connected to process performance such as the Size of the Product (Lines of Code) developed within a certain time frame, Product Quality (Ratio of Test Cases Passed, Defect Removal Efficiency) etc.

The double arrows in Figure 2.4 depict the probable causal relationships between structural and process performance via the software development process model implemented. Similarly, in the study described by Harter et al (2000), the level of maturity in SD (in CMMI) was positively correlated with product quality.

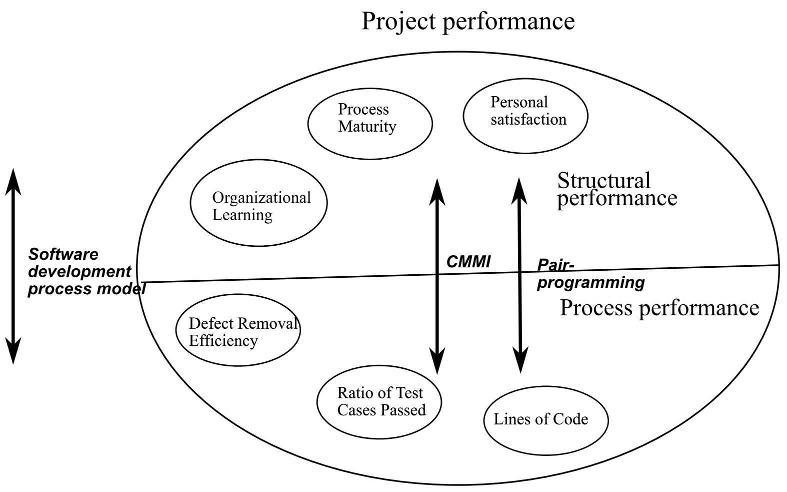


**Figure 2.4.** Project performance outline

Source: Composed by the author

In the SD process, the interaction between process and structural performance is implemented via different SD process evaluation and development methodologies as SD process models. The SD process model, in the sense of combining the concepts of structural and process performance, is an organized set of rules and structures designed to perform the SD process in order to satisfy the management, customer and project team needs simultaneously.

Some authors (Jiang et al 2004) divide process performance into efficiency and effectiveness. "Efficiency is often considered to be measured by the quality of the software product, adherence to budgeted time and money, and cost of the software operation. Effectiveness is considered to be the applicability and adaptability of the software." (ibid: 281).

Although process performance can be interpreted differently, the improvement of process performance proceeds via structural performance. This position is explained clearly by the developer participating in the Linbergs' (1999: 188) study – "Although the project focus gives projects much empowerment, this arrangement is no longer appropriate for a complex product-line like we have. There are far too many inter-dependencies. Competition ends up delaying our products or making them less successful. A better structure that supports collaboration is needed".

The same idea is supported by Jiang et al (2004: 281) as "...the organizational issues involve the knowledge gained by the organization during development, the interpersonal relations maintained, and the ability to control the resources used by the project". This is evidence that process performance cannot be improved independently of structural performance.

# 3. STRUCTURAL PERFORMANCE

This section presents the results of the author's studies in terms of structural performance. At the cellular level and in the controlled environment explored, structural performance is described in *Study I.* Structural performance in the actual SD environment as an OL pattern is presented in *Study II.* A qualitative structural performance evaluation methodology, derived from CMMI, is introduced in *Study III.*

## Structural performance at the cellular level

Structural performance at the cellular level is explored in *Study I* focusing on pair-programming – a technique used in agile methodologies, where two programmers work simultaneously on the same task. Several previous studies (Williams 2000; Williams and Kessler 2000b), have mostly demonstrated the emotional and satisfaction related advantages of pair-programming. The current research aims to discover the differences in performance using non-pair- and pair-programming teams. The research method implemented is a controlled experiment. The rationale behind this approach is that, although two programmers are working together on the same task and are therefore expending more hours than if programming alone, the quality of the teamwork is higher. Two different structures of team were implemented in the experiment: 1) two workers in a team working separately on different tasks (non-pair programming), 2) two workers working together in a team on the same task (pair-programming). These two structures also provide two different process performance characteristics – in the case of non-pair programming, process performance seems to be greater because the two programmers can perform separate tasks. Although direct process performance (for example, number of produced Lines of Code) is lower in the case of pair-programming, the quality of the delivered code (process performance in general) is greater and needs less amendments or improvements afterwards; for example, during the testing and debugging phase.

The advantages of code delivered using a pair-programming team do not appear in quantitative measures of productivity per employee, but evidently the teamwork appears to produce higher structural performance creating higher SD process performance. The main result of *Study I* is that in terms of direct process performance the efficiency of pair-programming is about the same as non-pair programming. But that additional organizational memory and greater team member satisfaction with 1) the result, 2) the method and 3) the partner does result from pair-programming.

## Structural performance and the OL environment

The non-pair vs pair-programming dimension characterizes unilateral structural performance at the cellular team level. To generalize the concept of structural performance at the project team level, a new framework is necessary. More diversified structural performance is explored at the SD team level on the basis of the OL environment. As presented in Figure 2.3, OL is a feedback-based cyclic process led by the management of the project team. OL appears in the SD process and is described in several ways. For example, Nonaka et al (2003) describes the OL environment as a "... diversified space, where organization members learn together, as well as the result of the OL process". An OL environment consisting of mental models, feelings, emotions and experiences as a mental, organizational and social pattern is described by Senge (1990) in the form of a five-factor model and by Mets (2002) as a three-factor model specially identifying role of main process (here SD) in OL. The same OL environment characteristics (mental models and shared vision) are introduced in the agile approach by Kruchten (2007) and Kollmann (2008).

*Study II* aimed to explore the OL environment as a pattern of OL features in the SD project. The research instrument was a questionnaire consisting of one part on OL and another on project management. The project management part was included to discover the possible mutual impact of the features of OL and project management. Patterns of OL were derived via an exploratory factor analysis.

The study was performed in Estonian SD teams, and three- and five-factor patterns were explored as structure performance models. It was discovered that project management issues are genuine for the OL environment. Consequently, the OL environment in Estonian SD teams mostly corresponds to the five-factor model and the three-factor model. One industry specific feature is presented in both models – a strong feature called "Desire for personal mastery". This feature is evidence of a perceived demand to improve the developers' own skills.

In *Study II* Senge's five disciplines and Mets' three-factor model of the OL environment were implemented and adjusted for the SD project teams as a structural performance framework. In addition to this framework, the qualitative evaluation of structural performance is necessary to evaluate the quality of the OL environment. This evaluation issue was resolved in *Study III* as presented in the next section.

## Structural performance evaluation

Structural performance explored according to the features of OL as presented in the previous section is not comparable across projects. To compare the structural performance of different projects the qualitative method is implemented.

To find such a measure, the different software process models were analyzed. The software development process assessment tools, like CMMI, ISO 9001, ISO/IEC 12207 and ISO/IEC 15505 were analyzed to find another more qualitative way to define the structural performance of an SD project. As a result of the analysis, CMMI process area categories were chosen to express the structure of SD processes as structural performance. This choice was made because in comparison with the others, the result of the assessment for each process area category is expressed as a qualitative measure – self evaluated maturity level. Besides, the use of CMMI as the basis of structural performance evaluation was confirmed by the study by Harter et al (2000), identifying a clear connection between maturity levels and software product quality. Harter et al (ibid) introduced modelling structural performance in terms of maturity level according to CMMI and process performance in terms of Lines of Code, Errors Repaired, Effort and so on. The results of this study (ibid) indicate that although enhancing process maturity needs additional (development) effort: the greater the maturity or the greater the structural performance, the shorter the time needed to develop the product and the better the process performance in general.

The maturity level, as a result of the assessment of the development process, is usually evaluated by certification bodies. An external audit identifies the practices predefined by CMMI and specifies a maturity level for the SD process accordingly. *Study III* adapts the main ideas and criteria of the method to evaluate the maturity of the project team across the process area categories replacing the external audit with an internal self-evaluation. Each process area was evaluated according to the level of the applicability of the particular process area in the project. The evaluations were then summarized for each process area category. As opposed to conventional CMMI, maturity level is evaluated across process area categories as the maturity level for each particular category.

In the process of the current study, six semi-structured interviews with experienced project managers were performed. The structure of the interviews consisted of three parts – 1) general data, 2) project success related data, and 3) SD and OL related data. Project success related data includes a self-evaluation of process areas in the project predefined by CMMI. In the current study the quality management issues were also analyzed – the most popular quality assessment standard among Estonian SD teams is ISO 9001. Regarding the relation between SD process self evaluation and ISO certification, the ISO certified projects turned out to be more balanced across process area categories.

The methodology developed in *Study III* allows us to analyze maturity in the project relatively inexpensively and therefore, to evaluate the structural performance of the project.

# 4. FINDINGS AND DISCUSSION

In *Study I* structural performance appears at the cellular level as pair- and non-pair programming. Pair-programming in the SD process requires additional effort since it reduces the number of tasks performed in a specific period – two workers work on the same task. Most experiments and studies of pair-programming described in the literature focus on the emotional and social aspects of teamwork. For example, Williams and Kessler (2000a, 2000b) and Williams, Kessler, Cunningham and Jeffries (2000) and Williams (2000) present the benefits of pair-programming for collaborative work, adhering to procedures and standards, satisfaction with the work process and so on. The current study confirms the presence of these advantages. Besides these social and emotional aspects, *Study I* proves via a controlled experiment that the process performance results in both pair-programming and non-pair programming teams were statistically equivalent. In other words the additional effort of applying pair-programming does not essentially reduce the process performance of the SD process in general.

Team satisfaction was additionally analyzed in *Study I*. Satisfaction was divided using a factor analysis into three different types of satisfaction: satisfaction 1) with result, 2) with method and 3) with partner. As opposed to other studies, *Study I* showed lower satisfaction when using the pair programming method. The reason for this difference is the complexity of the tasks used in the experiment and dissatisfaction increased along with the complexity of the assignments. Dissatisfaction with result was transferred to dissatisfaction with method. A similar transference was described by McDowell, Werner, Bullock and Fernald (2003). McDowell et al (ibid.) also refer to satisfaction with result as *confidence*, and this is related to the "Desire for personal mastery" as a feature of the OL environment in *Study II*.

A more detailed and diverse example of structural performance was presented in *Study II* as the OL environment. A theoretical framework based on Senge's five disciplines and Mets' three-factor model was implemented and adjusted for the SD project teams. The number of features (in patterns) and the structure of the pattern turned out to be relevant to the theoretical basis of the study and also according to features of the OL environment discovered in agile environments. Mental models and shared vision are mostly presented in agile environments.

Compared to other studies exploring the structure of the OL environment in SD project teams, full correspondence between the features of the current empirical model was not established. The features described by Kelly (2003) do not correspond, but among the features defined by Shepherd et al (2006), partially corresponding features can be found in the empirical five-factor model.

In comparison with other Estonian industries, empirical models of Estonian SD teams are more accurate in terms of the orthogonality of the features. The correlations between the features in the three-factor empirical model are remarkably lower (between 0.06 and 0.14, on significance level 0.05 ) than in a similar model based on data from service and manufacturing enterprises (between 0.22 and 0.42 accordingly, on significance level 0.001) (Mets and Torokoff 2007).

Among the OL features presented in the Estonian SD teams “Desire for personal mastery” appeared to be a strong feature. This feature was similarly presented in the five-factor model and the three-factor model as well. According to the analysis of team satisfaction in *Study I* dissatisfaction with results was similarly strong for complicated assignments. These two findings are evidence of the perceived demand to improve their own skills among Estonian project team members.

To evaluate the quality of the structural performance, the framework for self-evaluation was created in *Study III*. This framework allows us to analyze the maturity in the company, and therefore, evaluate the structural performance in the project team relatively inexpensively. The actual SD maturity evaluated among Estonian software developing companies in the current study is not very high – only basic process areas are evident. The motivation for software process improvement increases team satisfaction in terms of structural performance: emotionally successful projects were more mature and balanced.

## Contribution of the thesis

SD project success is the consensus of different project stakeholders. This creates the need for appropriate methodologies and concepts. In this work we have explored structural performance improvement as a managerial tool for process performance enhancement. Structure performance is a suitable concept for building consensus between project stakeholders’ interests for SD project success.

The theoretical value of the thesis is its attempt to link the cellular level, organizational level and structural performance in software development. The OL environment explored among Estonian SD teams provided that link. The OL features explored in SD are clearer and statistically more independent than among other Estonian service and manufacturing companies. This confirms the viewpoint based on OL theory, that features of OL are more essential for knowledge intensive industries such as SD.

The practical outcomes of the current work include the OL features questionnaire and methodology for SD process maturity self-evaluation in SD teams. The SD process maturity self-evaluation methodology created in the process of the current study allows us to evaluate the quality of structural performance for internal needs without a costly and time-consuming certification process.

Instances of structural performance, presented in the current study, are provisional and serving the basis for following studies. Probably there exists a wide range of different useful frameworks of structural performance. Therefore the structural performance in general is a useful managerial tool for resolving SD development discrepancies and achieving SD project success.

# 5. CONCLUSIONS

An SD project usually has three main groups of stakeholders – the development company management, the project customer as software user and the project team as the software implementer. According to former studies, the interests of these groups differ. A successful project results when software that is user-friendly and works is delivered on time and within budget. Hence, project success is defined in terms of a consensus between the interests of these three groups of project stakeholders.

Project performance on the management level is important in relation to time and budget. This performance is a feature of the company since it is dealing with turnover and profit issues. However, project performance has meaning as a measure of success at the project level as well. Project performance is divided into process and structural performance. Process performance is more easily measured, but difficult to improve. Evaluating structural performance is more complicated. But in contrast to process performance, the improvement of structural performance is a task for the management.

The current research focuses on structural performance as an instrument of the project management to improve process performance, and therefore, to complete SD projects successfully. Firstly, structural performance is introduced at the cellular level as process performance in two different structures – pair- and non-pair programming. A more advanced instance of structural performance is the OL environment. Clearly identified OL features are evidence of agility in the project environment and a high level of structural performance. To evaluate structural performance more precisely, the concept of maturity levels is introduced. In addition to the conventional concept of maturity levels (CMMI), the adjusted self-evaluation is quicker and easier to use.

**Conclusion 1:** The analysis of the concepts presented in various studies shows that structural and process performance are important success factors for a software project. This PhD thesis mainly focuses on the social, organizational and managerial aspects, including organizational learning and project management. During the course of this research it became apparent that the performance of a project consists of a structural component (in the wider sense) and a dynamic component. Structural performance can be expressed in different ways, such as 1) the maturity of the development process, 2) organizational learning ability, and 3) the satisfaction of project team members. The structural performance of the development process can be measured according to its maturity level, which is evaluated by an independent certifier or by the project team via self-evaluation. Organizational learning ability expresses structural performance on the scale of "learning – non-learning". The third way of describing structural performance – project team satisfaction – can be divided into: 1) satisfaction with the result – motivational aspect, 2) satisfaction with the method (of work) – organizational aspect, and 3) satisfaction with the partner – social aspect.

**Conclusion 2:** Agile methodologies have been very popular since the publication of the Agile Manifesto in 2001, and they exploit relatively frequent use in small and medium sized businesses. One of the best known agile methodologies is pair programming. Structural performance can be observed particularly well in pair programming, since pair and non-pair programming are two clearly distinguishable work methods or structures. Most advantages of pair programming established in previous studies have been effectiveness-oriented and social in nature. These have also been confirmed by current research. In general, pair programming is regarded as more efficient (productive). Previous studies have focused more on this programming technique's social satisfaction related aspects, than the productiveness of the development process. The controlled experiment conducted during this research focused on measuring the productivity and no differences in productivity were observed for pair and non-pair programming. Nevertheless, these results showed that pair programming is not less productive and is thus useful as another technique to increase the process performance of the development process as a whole.

**Conclusion 3:** The agile approach as a learning environment in general is not so clearly explored as in the case of pair programming. Previous studies refer to the characteristics of the learning environment primarily in more intelligent and agile organizations. The best known is Senge's five-factor model of the organizational Learning. As a generalization of Senge's OL concept, Mets' three-factor model has been derived from the example of Estonian businesses. In the present study, project management characteristics inherent in software development were added to these approaches and a questionnaire for software companies was composed. As a result of the survey, OL characteristics for software companies were established in accordance with Senge's five-factor as well as the three-factor model previously observed in Estonian businesses. The first model also included modified characteristics of "Systems thinking" and "Mental models", the remaining components – "Work values", "Team development", and main process related "Desire for personal mastery" – were similar for both models. In giving importance to personal mastery, the need to improve one's skills felt among Estonian software developers can be noted. At the same time, even in a survey with a relatively small sample, the OL characteristics/ patterns in Estonian software development are more visible than in manufacturing and services organizations based on previous studies. This confirms our view of software development organizations as learning organizations.

**Conclusion 4:** Evaluation and development methodologies used in projects usually depend on the project size, market needs and level of maturity. Research so far has demonstrated that methodologies are the more formalised the larger the projects and the higher the level of maturity. Customer demand and the need to better organize the work of the development team are also among the main reasons for implementing formalised methods, such as CMMI and ISO9001, for evaluating development processes. Hence, evaluating partners based on formalised methodologies is used in global and especially in outsourced development services; often the certification of the development team is required. Our research in Estonia established that due to the lack of customer demand, certification is not used and most development projects are not very large. Consequently, the formalization of the development process remains low and various iterative and agile approaches are popular even in relatively large businesses, such as the IT department of one of the major banks active in Estonia.

**Conclusion 5:** A mature software development project is characterized by properly documented process performance data. Post-project interviews revealed that collecting process performance data is not a priority among Estonian developers – data for evaluating the quality of the software created existed for two projects out of six; data collection was considered complicated. Data about the volume of the project existed in two thirds of the projects researched. The situation as a whole shows Estonian software development being on half way to maturity in terms of project management and this implies unused opportunities for Estonian companies in the global knowledge economy.

# 6. LIMITATIONS AND FUTURE RESEARCH

Structural performance issues in the current dissertation have been explored at elementary (minimally two members) team level *(Study I),* project team level *(Study II)* and evaluated across process area categories as process maturity level *(Study III).*

The *Study I* was a carefully planned and performed pair-programming experiment in a controlled environment, but the essential limitation of this study was the sample. The first part of the study was performed among students, who do not have long-term experience in SD. Therefore, the generalization of results for experienced developers is somewhat problematic. However, the study demonstrated the potential for the rigorous analysis of SD process performance.

The sample in the second part of the study was more general, since experienced developers participated in this study. This sample was biased according to the fields of activity because more than half of the respondents were from software development department in Estonia's largest bank. The rest of the respondents were mostly from small and micro-sized SD companies.

*Study II* explores structural performance as features of OL. OL is significant at all levels of management – operational, tactical and strategic. The current study focused on projects as first of all an operational entity of management. Therefore, the aspects related to features of OL on tactical and strategic levels are not covered in this study.

The methodology for evaluating SD process maturity using self-evaluation developed in *Study III* is in its initial stages. The six interviews were the first attempt to explore the structural performance self-evaluation quantitatively. Hence, the future justification and generalization of the self-evaluation methodology would be necessary.

The general limitations for all studies performed are 1) locality, as only Estonian teams participated in the studies and 2) the diversity of the methodologies. Due to the long period and different methods of study, the process of joining and generalizing the collected results and some interpretations caused a slight bias because of the changing business environment and accumulation of experience by software developers in that period.

There is also no proven clear causal relationship between different more complicated structures and process performance. Pair-programming is a very simple example and future studies should prove or disprove the influence of different structures on performance.

The studies presented in this thesis can be considered as introductory research into structural performance. In the future, the challenge would be to repeat similar studies among the international community of software developers and across a full-scale range (from micro-sized to large) of SD enterprises to generalize the results. Also, some future research would be desirable to analyze more precisely the role of different structures in SD project success. It is reasonable as well to find more influential and convenient structures beside process assessment standards or Organizational Learning environments for managing SD process performance as a success factor of SD projects.

The list of instances of structural performance presented in the current study is not exhaustive and can be complemented in future studies. Structural performance as a phenomenon remains open to continuing interpretations in the future. Quite possibly, some other concepts and approaches exploring the structural performance characteristics also exist. For example, an approach based on project manager leadership is probably useful in small SD teams and might complement the structural performance description.

# REFERENCES

1. Argyris, C. (1976). Single-loop and double-loop models in research on decision making, *Administrative Science Quarterly*, Vol. 21, pp. 363–375
2. Cao, L., Ramesh, B. (2007) Agile software development: ad hoc practices or sound principles, *IEEE Pro* (Mar.–Apr. 2007), pp. 41–47
3. CMMI (2006), *CMMI for development, Version 1.2*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA
4. Cockburn, A. (2002) *Agile software development.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA
5. Delaney, J., T., Huselid, M. A. (1996) The impact of human resource management practices on perceptions of organizational performance, *The Academy of Management Journal*, Vol. 39, No. 4, pp. 949–969
6. Eveleens, J. L., Verhoef, C. (2010) The rise and fall of the Chaos report figures, *IEEE Software,* Vol. 27, No. 1
7. Faraj, S., Sproull, L. (2000) Coordinating expertise in software development teams, *Management Science,* Vol. 46, No. 12, pp. 1554–1568
8. Feldt R., Angelis, L., Torkar, R., Samuelsson, M. (2010) Links between the personalities, views and attitudes of software engineers, *Information and Software Technology*, Vol. 52, pp. 611–624
9. Fichman, R. G., Kemerer, C. F. (1997) The assimilation of software process innovations: an organizational learning perspective, *Management Science,* Vol. 43, No. 10 pp. 1345–1363
10. Fowler, M., Highsmith, J. (2001) Agile manifesto, *Software Development,* Vol 9, Part 8, pp. 28–35
11. Garcia-Morales, V. J., Llorens-Montes, F. J., Verdu-Jover, A. J. (2008). The effects of transformational leadership on organizational performance through knowledge and innovation, *British Journal of Managament*, Vol. 19, pp. 299–319
12. Glazer, H., Dalton, J., Anderson, D. J., Konrad, M., Shrum, S. (2008) CMMI or Agile: why not embrace both!, *Technical Note*, CMU/SEI-2008-TN-003
13. Gobeli, D. H., Koenig, H. F., Bechinger, I. (1998) Managing conflict in software development teams: a multilevel analysis, *Journal of Product Innovation Management*, Vol. 15, Issue 5, pp. 423–435
14. Guest, D., N. Conway, N., Dewe, P. (2004) Using sequential tree analysis to search for bundles of HR practices, *Human Resource Management Journal*, Vol. 14, No. 1, pp. 79–96
15. Harter, E. D., Krishnan, S. M. and Slaughter, A. S. (2000), Effects of process maturity on quality, cycle time, and effort in software product development, *Management Science*, Vol. 46, No. 4, pp. 451–66
16. Hoffman, H., F. and Lehner, F. (2001) Requirements engineering as a success factor in software projects, *IEEE Software,* Vol. 18, Issue 4, pp. 58–66
17. Huselid, M. A. (1995) The impact of human resource management practices on turnover, productivity and corporate financial performance, *The Academy of Management Journal*, Vol. 38, No. 3, pp. 635–672
18. Ibert, O. (2004) Projects and firms as discordant complements: organisational learning in the Munich software ecology, *Research policy*, Vol. 33, pp. 1529–1546

19. Jiang, J. J., Klein, G., Hwang H., Huang, J., Hung, S. (2004) An exploration of the relationship between software development process maturity and project performance, *Information and Management*, Vol. 41, pp. 279–288
20. Johnson, J. R. (2002) Leading the learning organization: portrait of four leaders, *Leadership & Organization Development Journal,* Vol. 23, No. 5, pp. 241–249
21. Kalermo, J., Rissanen, J. (2002) *Agile software development in theory and practice*, Master Thesis, University of Jyväskylä
22. Kelly, A. (2003) *Software development as organizational learning*, MBA thesis, Nottingham University Business School
23. Kivipõld, K., Ahonen, M. (2011) Evidence of relationship between organizational leadership capability and job satisfaction: the case of IT service organization in Estonia, In *Proceedings of the V International Conference Management Theory and Practice: Synergy in Organisations, Tartu, 01–02 April 2011*, (Editors) Vadi M., Jaakson, K., Kindsiko E.
24. Kollmann, J. (2008) *Designing the user experience in an agile context,* Project report, Faculty of Life Sciences, University College London
25. Kruchten, P. (2007) Voyage in the agile memeplex, *ACM Queue*, Vol. 5, No. 5, pp. 38–44
26. Kähkönen, T., Abrahamsson P. (2004) Achieving CMMI level 2 with enhanced extreme programming approach, *In Proceedings of PROFES*, Editors F. Bomarius and H. Iida, LNCS 3009, pp. 378–392
27. Lai, K. H., Cheng, T. C. E. (2005) Effects of quality management and marketing on organizational performance, *Journal of Business research* Vol. 58, Issue 4, pp. 446–456
28. Linberg, K. R. (1999) Software developer perception about software project failure: a case study, *The Journal of Systems and Software*, Vol. 49, pp. 177–192
29. McDowell, C., Werner, L., Bullock, H. E., Fernald, J. (2003) The impact of pair programming on student performance, perception and persistence, *25th International Conference on Software Engineering*, Portland, Oregon, May 03–May 10
30. Mathiassen, L., Pourkomeylian, P. (2003) Managing knowledge in software organization, *Journal of Knowledge Management*, Vol. 7, No. 2, pp. 63–80
31. Mets, T., Torokoff, M. (2007) Patterns of learning organization in Estonian companies, *TRAMES. A Journal of the Humanities and Social Sciences*, Vol. 11, No. 2, pp. 139–154
32. Mets, T. (2002) Learning-based strategic development framework: implementation in Estonian company, *Management of Organizations: Systematic Research*, No. 23, pp. 83–93
33. Misra, S. C., Kumar, V., Kumar, U. (2009) Identifying some important success factors in adopting agile software development practices, *The Journal of Systems and Software*, Vol. 82, pp. 1869–1890
34. Mitki, Y., Shani, A. B., Meiri, Z. (1997) Organizational learning mechanisms and continuous improvement: a longitudinal study, *Journal of Organizational Change Management,* Vol. 10, No. 5, pp. 426–446
35. Nidomulu, S. R. and Subramani, M. R. (2003) The matrix of control: combining process and structure approaches to managing software development, *Journal of Management Information Systems*, Vol. 20 No. 3, pp. 159–96
36. Nonaka. I., Toyama, R., Byosiere, P. (2003) A theory of organizational knowledge creation: understanding the dynamic process of knowledge, *Handbook of Organizational Learning & Knowledge*, Oxford University Press, New York, pp. 491–517

37. Petter, S. (2008) Managing user expectations on software projects: Lessons from the trenches, *International Journal of Project Management,* Vol. 26, Issue 7, pp. 700–712
38. Pikkarainen, M., Salo, O., Still, J. (2005) Deploying agile practices in organisations: a case study, in *Proceedings of Euro SPI*, Editors I. Richardson, et. al., *Lecture Notes of Computer Science*, Vol. 3792, pp. 16–27
39. Procaccino, J. D., Verner, J. M., Overmyer, S. P., Darter, M. E. (2002) Case study: factors for early prediction of software development success, Information and Software Technology, Vol. 44, pp. 53–62
40. Procaccino, J. D., Verner, J. M., Shelfer K. M., Gefen, D. (2005) What do software practitioners really think about project success: an exploratory study, *The Journal of Systems and Software*, Vol. 78, pp. 194–203
41. Procaccino, J. D., Verner, J. M., Darter, M. E., Amadio, W. (2005) Toward predicting software development success from the perspective of practitioners: an exploratory Bayesian model, *Journal of Information Technology*, Vol. 20, No. 3, pp. 187–200
42. Qumer, A., Henderson-Sellers, B. (2010) Empirical evaluation of the agile process lifecycle management framework, in *Proceedings of Fourth International Conference on Reserch Challenges in Information Sciences*, editors Loucopoulos, P., Cavarero, J. L., Nice, May 19–21, France, pp. 213–222
43. Rifkin, W., Fulop, L. (1997) A review and case study on learning organizations, *The Learning Organization,* Vol. 4, No. 4, pp. 135–148
44. Rising, L., Janoff, N. S. (2000) The Scrum software development process for small teams, *IEEE Software,* Vol. 17, Issue 4, pp. 26–32
45. Royce W. W. (1987) Managing the development of large software systems: concepts and techniques, in *Proceedings of the 9th international conference on Software Engineering*, Monterey, California, United States, pp. 328–338, IEEE Computer Society Press
46. Salo, O., Pikkarainen M. (2005) *Agile deployment model*, public report, Agile VTT, Finland
47. Senge, P. (1990) *The fifth discipline. The art and practice of the learning organization*, Random House, Sydney
48. Shepherd M. M., Tesch B. D., Hsu S. C. J. (2006) Environmental traits that support a learning organization. The impact on information system development projects, *Comparative Technology Transfer and Society*, Vol. 4 No. 2, pp. 196–218
49. Tomaszewski, P., Lundberg, L. (2005) Software development productivity on a new platform: an industrial case study, *Information and Software Technology*, Vol. 47, pp. 257–269
50. Tseng Y., Lee T. (2009) Comparing appropriate decision support of human resource practices on organizational performance with DEA/AHP model, *Expert Systems with Applications*, Vol. 36, Issue 3, Part 2, pp. 6548–6558
51. Turner (2002), Agile development: good process or bad attitude, *Lecture Notes in Computer Science*, Vol 2559, pp. 134–144
52. Williams, L., Kessler, R. R. (2000a) All I really need to know about pair programming I learned in kindergarten, *Communications of the ACM*

53. Williams, L., Kessler, R. R. (2000b) Experimenting with industry's "Pair-Programming" model in the computer science classroom, *Journal on Software Engineering Education*
54. Williams, L., Kessler, R. R., Cunningham, W., Jeffries, R. (2000) Strengthening the case for pair programming, *IEEE Software*
55. Williams, L. (2000) *The Collaborative Software Process*, University of Utah

# ACKNOWLEDGEMENTS

The author is grateful to his supervisors, Tõnis Mets and Jüri Kiho, for useful advices, help and support, always encouraging him whenever he needed it. He would also like to thank his co-authors and all his other colleagues and friends for supporting him throughout his long and finally successful PhD studies. He is very thankful to his family, especially to wife Eve, for all this long years of understanding and support.

The author has been partly financially supported by the Estonian Ministry of Education's project SF0180037s08, by the Centre for Entrepreneurship and Institute of Computer Science of University of Tartu. He is also thankful to his former employer Cybernetica AS, one of the few private research companies in Estonia, where the initial idea of this study was born. Last but not least, the author would like to thank numerous students, developers and other software development team members, who contributed to data collection used in this study. Printing the dissertation has been supported by European Social Fund.

# SUMMARY IN ESTONIAN

## Struktuurne tulemuslikkus tarkvaraprojekti edutegurina – Eesti kogemus

Tarkvaraarenduse projekt on sama vana mõiste kui tarkvaraarendus ise – esimesed tarkvaraarendusprojektid viidi läbi juba eelmise sajandi kuuekümnendatel aastatel. Pikale ajaloole vaatamata on tarkvaraprojektide edukuse protsent jätkuvalt madal. Sellise olukorra üheks põhjuseks on asjaolu, et tarkvaraprojekti edukus võib olla määratletud mitmel erineval viisil. Näiteks konkreetne tarkvaraprojekt on edukaks ettevõtte juhtkonna jaoks, kui vajalik funktsionaalsus on valminud tähtajaks ja selle valmimiseks tehtud kulutused on jäänud eelarve piiresse. Süsteemi loomisel osalenud arendusmeeskonna vaade sellele samale projektile võib aga olla hoopiski erinev. Sageli väärtustavad projektimeeskonna liikmed lisaks teostatud funktsionaalsusele ka projekti käigus esinenud tehnoloogilisi väljakutseid, saadud kogemusi ja uusi oskusi. Samuti on olulisel kohal projektimeeskonna liikmete rahulolu. Seega saame öelda, et sageli on projekti edukuse kriteeriumid erinevate projekti osapoolte jaoks erinevad.

Kui ettevõtte juhtkonnale oluliste edutegurite hulgas on tähtaegadest ja eelarvest kinni pidamine selgelt ja üheselt määratletav, siis kolmas edutegur – ärilistele eesmärkidele vastavus – on keerulisem. Lisandub kolmas osapool – loodava tarkvara kasutaja. Projekti edukust saame seega vaadelda kolme erineva osapoole – tarkvara loonud projektimeeskonna, tarkvarafirma juhtkonna ja tarkvara kasutaja – konsensusena. Edukas tarkvaraprojekt on konsensus kõigi nende kolme osapoole huvide vahel.

Projekti edukusega on tihedalt seotud teine mõiste – projekti tulemuslikkus. Üldiselt, kui räägitakse projekti tulemuslikkusest, siis mõeldakse selle all projektimeeskonna tulemuslikkust. Projekti parem tulemuslikkus annab projektile paindlikkuse ja võimaldab seeläbi jõuda väiksema ressursikuluga projekti edukaks lõpetamiseks vajaliku konsensuseni.

Projekti tulemuslikkus jaguneb struktuurseks tulemuslikkuseks ja protsessi tulemuslikkuseks. Esimest korda on selliselt kahte liiki projekti tulemuslikkust eristanud Niddomulu ja Subbaramani (2003). Struktuurne tulemuslikkus väljendab projektimeeskonna struktuuri „ ... kirjeldades projektis kehtivaid standardeid ja valikuid otsuste delegeerimiseks”. Protsessi tulemuslikkus kirjeldab „ ... talitusviise ja tulemeid kui võtmemeetmeid projekti töö juhtimiseks” (ibid).

Nii ettevõtte kui ka projektimeeskonna tulemuslikkuse aluseks on töötajate innovatiivsus, nende võimed ja oskused ning motivatsioon, samuti omavahelised suhted ehk kokkuvõtlikult – töötajaskonna võimekus. Siiski ei piisa tulemuslikkuseks üksnes vajaliku võimekuse olemasolust. Selleks, et võimekusest saaks konkreetne tulemus on tarvis veel midagi, mis viiks projektimeeskonna potentsiaali või meeskonna võimekuse konkreetse tulemuseni ehk siis realiseeriks võimekuse tulemuslikkusena. Oluline on teada projektimeeskonna eriomadusi, mis on vajalikud kogu meeskonna võimekuse realiseerumiseks ja seeläbi projekti eduks. *Doktoritöö eesmärk on identifitseerida tarkvaraprojekti eriomadused, mis on aluseks projektimeeskonna struktuursele tulemuslikkusele, ning kirjeldada ja üldistada struktuurse tulemuslikkuse mudelid Eesti projektimeeskondade näitel.*

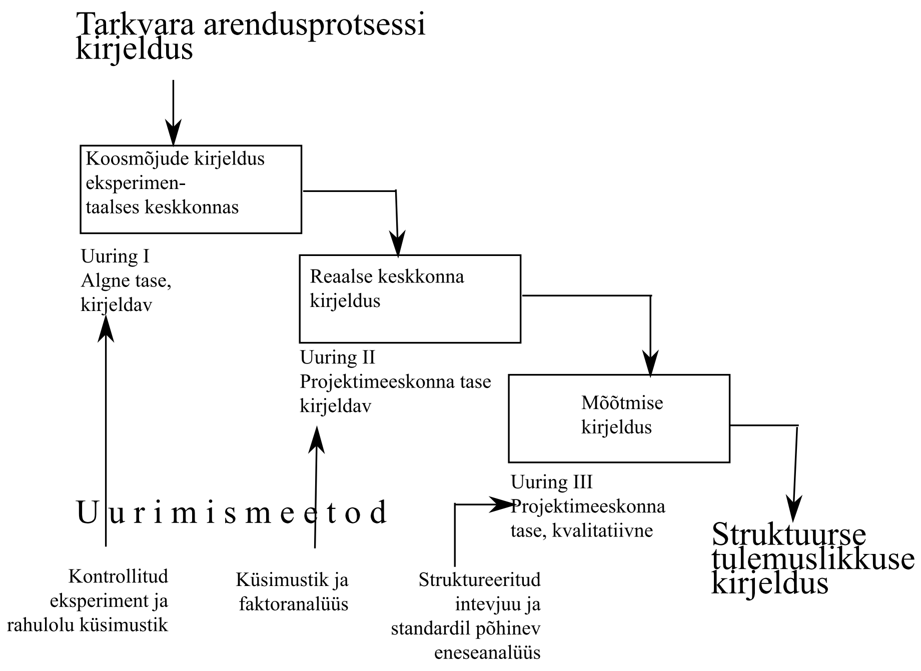
Doktoritöö koosneb kolmest suhteliselt erinevast kuid kontseptuaalselt seotud Eesti tarkvara-arendusmeeskondade hulgas läbi viidud uuringust. Töö põhiliseks eesmärgiks on uurida struktuurse tulemuslikkuse olemust ja kirjeldada struktuurse ning protsessi tulemuslikkuse vahelist seost. Kõigepealt kirjeldatakse struktuurset tulemuslikkust *Uuringus I* algsel minimaalsel tasemel, kus projektimeeskond koosneb kahest isikust. Selle eksperimendi eesmärgiks on kirjeldada seoseid struktuurse (paaris-, mittepaaris-programmeerimine) ja protsessi tulemuslikkuse (tootlikkuse) vahel. Uurimuse käigus oli plaanis tuvastada, kas meeskonna struktuur mõjutab programmeerimise tulemuslikkust (tootlikkust). Uurimismeetodiks oli kontrollitud eksperiment. Samas uurimuses on kirjeldatud ka minimaalset projektikeskkonda kui eksperimendis osalejate rahulolu, mille mõõtmiseks on kasutatud vastavat küsimustikku.

Selleks, et tuvastada tarkvaraarendusprojekti keskkonna kui struktuurse tulemuslikkuse määra eripära mitte-eksperimentaalses olukorras, on läbi viidud *Uuring II.* Kuna tarkvaraprojekti keskkond on tegelikus tarkvaraarenduses oluliselt mitmetahulisem, siis on selle uurimuse taristuks valitud õppiva organisatsiooni kontseptsioon. Uurimismeetodiks on õppiva organisatsiooni küsimustik ja sellele järgnev faktoranalüüs, mille abil lisaks õppiva organisatsiooni omadustele on uuritud ka tarkvaraprojekti juhtimisega seotud eripärasid projektimeeskonnas. Kuna projektimeeskonda pole võimalik vaadelda lahus ettevõttest, milles projektimeeskond töötab, siis on selles uurimuses vaatluse all ka ettevõtte töökorraldusega seotud aspektid nagu töötajate koolitus, projektide plaanimine jms.

Selleks, et saada täpsemat ettekujutust struktuurse tulemuslikkuse hindamisest, eriti praktilistes rakendustes, on läbi viidud *Uuring III,* mille käigus mõõdeti struktuurse tulemuslikkuse taset kui CMMI arendusprotsessi küpsuse taset. Uurimismeetodiks on selles uurimuses struktureeritud intervjuud, mille käigus uuriti Eesti tarkvaratootjate kuut projekti.

Uurimisprotsessi käik ja uuringute omavaheline seos on kujutatud Joonisel 1.

Uurimisprotsessi lõpptulemusena valmis struktuurse tulemuslikkuse kirjeldus kui õppiva organisatsiooni keskkonna kirjeldus tarkvara projektimeeskonna jaoks. Seda keskkonda kirjeldab õppiva organisatsiooni nii kolme- kui ka viiefaktoriline mudel. Kvalitatiivselt iseloomustab projektimeeskonna struktuurset tulemuslikkust CMMI küpsustase, mis on hinnatud üle kõikide CMMI protsessivaldkondade.


Tarkvara arendusprotsessi kirjeldus
Koosmõjude kirjeldus eksperimentaalses keskkonnas
Uuring I
Algne tase, kirjeldav
Reaalse keskkonna kirjeldus
Uuring II
Projektimeeskonna tase kirjeldav
Mõõtmise kirjeldus
Uuring III
Projektimeeskonna tase, kvalitatiivne
Struktuurse tulemuslikkuse kirjeldus
Uurimismeetod
Kontrollitud eksperiment ja rahulolu küsimustik
Küsimustik ja faktoranalüüs
Struktureeritud intevjuu ja standardil põhinev eneseanalüüs

**Joonis 1.** Uurimisprotsessi kirjeldus

Allikas: Autori koostatud

Juhtimisülesandena pole võimalik otseselt protsessi tulemuslikkust parandada, näiteks suurendada otseselt edukalt läbitud testide arvu. Seda on võimalik teha vaid struktuurse tulemuslikkuse suurendamise ehk projekti keskkonna parandamise kaudu. Projektimeeskonna parem struktuurne tulemuslikkus (näiteks CMMI või mõne muu otstarbekalt ja süstemaatiliselt rakendatud hindamis- või arendusmetoodika mõttes) tagab ka parema protsessi tulemuslikkuse.

Siiski kulutab struktuurse tulemuslikkuse suurendamine projekti jaoks eraldatud ressursse, seega jääb projekti otsese edenemise jaoks neid vähem. Projekti õnnestumise tõenäosust suurendab projekti struktuurse tulemuslikkuse parandamine vaid sel juhul, kui selle läbi saadav aja kokkuhoid on suurem kui struktuurse tulemuslikkuse parandamiseks kulunud aeg.

Järgnevalt on esitatud töö põhijäreldused.

1. Doktoritöös läbi viidud uuringute käigus selgus, et projekti tulemuslikkus koosneb struktuursest komponendist (laias mõttes) ja dünaamilisest komponendist. Struktuurne tulemuslikkus on väljendatav mitmel erineval viisil nagu 1) arendusprotsessi küpsustase, 2) organisatsioonilise õppimise võimekus ja 3) projektimeeskonna liikmete rahulolu. Arendusprotsessi struktuurset tulemuslikkust saab mõõta kui arendusprotsessi küpsustaset, mis on hinnatud

sõltumatu sertifitseerija poolt või siis projektimeeskonna poolt eneseanalüüsina. Organisatsioonilise õppimise võimekus väljendab struktuurset tulemuslikkust „õppiv vs. mitteõppiv" skaalal. Kolmas viis struktuurse tulemuslikkuse väljendamiseks – projektimeeskonna rahulolu, on jaotatav 1) rahuloluks tulemusega – motivatsiooniline aspekt, 2) rahuloluks (töö) meetodiga – organisatsiooniline aspekt ja 3) rahuloluks partneriga – sotsiaalne aspekt.

2. Struktuurne tulemuslikkus on iseloomulik ka alates 2001 aastast laiemalt kasutusele võetud agiilsetele *(agile)* metoodikatele. Eriti selgelt on struktuurne tulemuslikkus väljendunud paarisprogrammeerimises, kuna paaris- ja mittepaaris-programmeerimine on kaks selgelt eristatavat tööviisi ehk struktuuri. Enamus varasemates uuringutes tuvastatud paarisprogrammeerimise eeliseid on olnud tõhususele orienteeritud ja sotsiaalse iseloomuga. Need erisused said kinnitust ka käesolevas uuringus. Üldiselt on arvatud, et paarisprogrammeerimine on tõhusam (produktiivsem). Eelmistes uurimustes on enam tähelepanu pööratud selle programmeerimistehnika sotsiaalsetele ja rahuloluga seotud aspektidele, kui arendusprotsessi tulemuslikkusele. Uuringu käigus läbi viidud kontrollitud eksperiment keskendus tulemuslikkuse mõõtmisele, kuid ei avastanud erinevusi paaris- ja mittepaarisprogrammeerimise tulemuslikkuse vahel. Siiski näitasid saadud tulemused, et paarisprogrammeerimine pole vähem produktiivne ja on seeläbi kasutatav kui üks tehnikatest, mis tõstab kogu arendusprotsessi tulemuslikkust.
3. Agiilne projektikeskkond üldiselt, kui organisatsioonilise õppimise keskkond, ei ole nii selgelt väljendunud kui paarisprogrammeerimise puhul. Varasemad uuringud viitavad õppimiskeskkonna tunnustele eelkõige intelligentsemates organisatsioonides. Enimtuntud on Senge õppiva organisatsiooni (ÕO) viiefaktoriline mudel. Üldistades Senge ÕO kontseptsiooni on Eesti ettevõtete näitel tuletatud Metsa kolmefaktoriline mudel. Täiendades käsitlust tarkvaraarendusele omaste projektijuhtimise tunnustega koostati käesolevas uuringus ÕO küsimustik tarkvarafirmadele. Küsitluse tulemusena tuvastati tarkvarafirmadele omased ÕO tunnused vastavuses nii Senge viiefaktorilisele kui ka varem Eesti ettevõtetes tuvastatud kolmefaktorilisele mudelile. Esimene neist eristus „jagatud visiooni" ja „mõttemudelite" modifitseeritud tunnustega, ülejäänud komponendid „tööväärtused", „meeskonna arendamine" ja põhiprotsessiga seonduv „isikliku meisterlikkuse vajadus" olid mõlemal mudelil sarnased. Isikliku meisterlikkuse tähtsustamises võib näha Eesti tarkvaraarendajate hulgas tajutud vajadust oma oskuste parandamiseks. Samas, isegi suhteliselt väikese valimiga uuringus on ÕO tunnused/ mustrid Eesti tarkvaraarenduses paremini välja joonistunud kui ülejäänud tootmis- ja teenindusorganisatsioonides varasemate uuringute põhjal. See kinnitab tarkvaraarendus-organisatsiooni iseloomustust õppiva organisatsioonina.
4. Projektides rakendatud hindamis- ja arendusmetoodikad sõltuvad tavaliselt projekti suurusest, turuvajadustest ja küpsustasemest. Senised uurimused on

näidanud, et metoodikad on seda formaliseeritumad, mida suuremate ja kõrgema küpsustasemega projektidega on tegemist. Kliendinõudlus, aga ka vajadus arendusmeeskonna töö paremaks korraldamiseks on samuti arendusprotsesside hindamise formaliseeritud metoodikate nagu CMMI ja ISO9001 rakendamise peamisteks põhjusteks. Seega partnerite hindamine formaliseeritud metoodika alusel on kasutusel globaalse ja eriti sisse ostetava arendusteenuse puhul, sageli nõutakse arendusmeeskonna (CMMI) sertifitseeritust. Meie uuring Eestis tuvastas, et kliendinõudluse puudumise tõttu pole CMMI kasutusel ning enamus arendusprojekte pole väga suured. Seetõttu on ka arendusprotsessi formaliseeritus madal ja populaarsed on erinevad iteratiivsed ja väledad lähenemised isegi suhteliselt suurtes ettevõtetes, nagu näiteks ühe Eestis tegutseva suure panga IT osakonnas.

5. Küpset tarkvaraarendusprojekti iseloomustavad korrektselt dokumenteeritud tulemuslikkuse andmed. Projektijärgsed intervjuud näitasid, et tulemuslikkuse andmete kogumine pole prioriteediks Eesti arendajate hulgas: andmed loodud tarkvara kvaliteedi hindamiseks olid olemas ainult kahe projekti puhul kuuest – põhjenduseks andmete kogumise keerukus. Andmed projekti mahu kohta olid olemas kahel kolmandikul uuritud projektidest. Situatsioon tervikuna iseloomustab Eesti tarkvaraarendust olevat poolel teel projektijuhtimise küpsusele ja viitab tööstusharu kasutamata võimalustele globaalses teadmusmajanduses.

## PUBLICATIONS

# CURRICULUM VITAE

| | |
|---|---|
| Name | Uuno Puus |
| Date of birth | 28.06.1956 |
| Citizenship | Estonia |
| Home | Kastani 24a–19, 50410 Tartu, Estonia |
| Phone | +372 7 376 362 |
| E-mail | uuno.puus@ut.ee |

**Current position**
Researcher, University of Tartu, Faculty of Economics  and Business Administration, Centre for Entrepreneurship

**Education**

| | |
|---|---|
| 2002 – Present | PhD student, University of Tartu; |
| 1999–2002 | MSc, Informatics, University of Tartu; |
| 1987–1993 | BSc, Psychology, University of Tartu; |
| 1974–1979 | Bsc, Applied mathematics, Tartu State University (University of Tartu today). |

**Employment History**

| | |
|---|---|
| 2010–Present, | Researcher, University of Tartu, Faculty of Economics and Business Administration, Centre for Entrepreneurship; |
| 2008–2010 | Consultant-advisor, University of Tartu, Faculty of Economics and Business Administration, Centre for Entrepreneurship; |
| 2002–2009 | Researcher, Cybernetica AS; |
| 1998–2009 | Head of Laboratory, Cybernetica AS; |
| 1994–1998 | Accountant, Financial analyst, Tartu Maja Ltd; |
| 1994 | IT specialist, L. M. R. A. Ltd; |
| 1992–1994 | IT specialist, Tartu County Government; |
| 1979–1992 | Director, Teacher, Nõo Secondary School. |

**Studies and practices**

| | |
|---|---|
| 2006, November | Project Management for Software Development, (1 week course), Tallinn, Cybernetica, Learningtree International, course No. 340; |
| 2006, September | Management and Audit of Software Development Projects by P. Ceulemans (2-day course), ISACA Estonian chapter, Tallinn |
| 2006, February | Managing Complex Projects, (1 week course), Tallinn, Cybernetica, Learningtree International, course No. 287; |
| 2006, January | Project Quality Management for Project Managers, (1 week course), Tallinn, Cybernetica, Learningtree International, course No. 349; |
| 2005 January | Innovative product commercialization, (2 month practice) Netherlands, Groeningen, Zernike group. |

# ELULOOKIRJELDUS

| | |
|---|---|
| Nimi | Uuno Puus |
| Sünniaeg | 28.06.1956 |
| Kodakondsus | Eesti |
| Kodune aadress | Kastani 24a–19, 50410 Tartu |
| Telefon | +372 7 376 362 |
| E-mail | uuno.puus@ut.ee |

**Amet, töökoht**
Teadur, Tartu Ülikool, Majandusteaduskond, Ettevõtluskeskus

**Haridustee**

| | |
|---|---|
| 2002–praeguseni, | Tartu Ülikool, doktorant; |
| 1999–2002 | Tartu Ülikool, MSc informaatikas; |
| 1987–1993 | Tartu Ülikool, BSc psühholoogias; |
| 1974–1979 | Tartu Riiklik Ülikool (praegune Tartu Ülikool) BSc rakendusmatemaatikas. |

**Teenistuskäik**

| | |
|---|---|
| 2010–praeguseni, | Teadur, Tartu Ülikool, Majandusteaduskond, Ettevõtluskeskus; |
| 2008–2010 | Konsultant-nõustaja, Tartu Ülikool, Majandusteaduskond, Ettevõtluskeskus; |
| 2002–2009 | Teadur, Cybernetica AS; |
| 1998–2009 | Laborijuhataja, Cybernetica AS; |
| 1994–1998 | Raamatupidaja, finantsanalüütik, Tartu Maja AS; |
| 1994 | IT spetsialist, L. M. R. A. AS; |
| 1992–1994 | IT spetsialist, Tartu Maavalitsus; |
| 1979–1992 | Direktor, õpetaja, Nõo Keskkool (praegu Nõo Reaalgümnaasium). |

**Täiendõpe ja stažeerimine**

| | |
|---|---|
| 2006, november | Project Management for Software Development, (nädalane kursus), Tallinn, Cybernetica, Learningtree International, kursus nr. 340; |
| 2006, september | Management and Audit of Software Development Projects, lektor P. Ceulemans (2-päevane kursus), ISACA Eesti haruühing, Tallinn |
| 2006, veebruar | Managing Complex Projects, (nädalane kursus), Tallinn, Cybernetica, Learningtree International, kursus nr. 287; |
| 2006, jaanuar | Project Quality Management for Project Managers, (1 week course), Tallinn, Cybernetica, Learningtree International, kursus nr. 349; |
| 2005, jaanuar | Innovative product commercialization, (2 kuuline stažeerimine) Holland, Groeningen, Zernike group. |

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
20. **Jüri Lember.** Consistency of empirical k-centres. Tartu, 1999, 148 p.
21. **Ella Puman.** Optimization of plastic conical shells. Tartu, 2000, 102 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
25. **Maria Zeltser.** Investigation of double sequence spaces by soft and hard analitical methods. Tartu, 2001, 154 p.
26. **Ernst Tungel.** Optimization of plastic spherical shells. Tartu, 2001, 90 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 p.
28. **Rainis Haller.** *M(r,s)*-inequalities. Tartu, 2002, 78 p.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
30. **Eno Tõnisson.** Solving of expession manipulation exercises in computer algebra systems. Tartu, 2002, 92 p.
31. **Mart Abel.** Structure of Gelfand-Mazur algebras. Tartu, 2003. 94 p.
32. **Vladimir Kuchmei.** Affine completeness of some ockham algebras. Tartu, 2003. 100 p.
33. **Olga Dunajeva.** Asymptotic matrix methods in statistical inference problems. Tartu 2003. 78 p.
34. **Mare Tarang.** Stability of the spline collocation method for volterra integro-differential equations. Tartu 2004. 90 p.
35. **Tatjana Nahtman.** Permutation invariance and reparameterizations in linear models. Tartu 2004. 91 p.
36. **Märt Möls.** Linear mixed models with equivalent predictors. Tartu 2004. 70 p.
37. **Kristiina Hakk.** Approximation methods for weakly singular integral equations with discontinuous coefficients. Tartu 2004, 137 p.
38. **Meelis Käärik.** Fitting sets to probability distributions. Tartu 2005, 90 p.
39. **Inga Parts.** Piecewise polynomial collocation methods for solving weakly singular integro-differential equations. Tartu 2005, 140 p.
40. **Natalia Saealle.** Convergence and summability with speed of functional series. Tartu 2005, 91 p.
41. **Tanel Kaart.** The reliability of linear mixed models in genetic studies. Tartu 2006, 124 p.
42. **Kadre Torn.** Shear and bending response of inelastic structures to dynamic load. Tartu 2006, 142 p.