2012 Eighth International Conference on the Quality of Information and Communications Technology

Experimental Software Engineering in Educational Context

Luís M. Alves
Escola de Tecnologia e Gestão
Instituto Politécnico de Bragança
Bragança, Portugal
lalves@ipb.pt

Ricardo J. Machado
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
rmac@dsi.uminho.pt

Pedro Ribeiro
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
pmgar@dsi.uminho.pt

***Abstract* — Empirical studies are important in software engineering to evaluate new tools, techniques, methods and technologies in a structured way before they are introduced in the industrial (real) software process. Within this PhD thesis we will develop a framework of a consistent process for involving students as subjects of empirical studies of software engineering. In concrete, our experiences with software development teams composed of students will analyze how RUP (Rational Unified Process) processes can be compliant with the CMMI (Capability Maturity Model Integration), namely in the context of MLs (maturity levels) 2 and 3. Additionally, we will also analyze the influence of project management tools to improve the process maturity of the teams. Our final goal of carrying out empirical studies with students is to understand its validity when compared with the corresponding studies in real industrial settings.**

Keywords: software engineering management, software engineering process, software quality

I. INTRODUCTION

In the early nineties, Basili introduced, for the first time, the concept of *experience factory*. As the author refers in [1] the concept was introduced to "institutionalize the collective learning of the organization that is at the root of continual improvement and competitive advantage". Thus, the *experience factory* provides an organizational schema for collecting experiences on reuse of empirical results, for analyzing them and generalizing the knowledge contained [2]. This scheme was designed based on many years of the Software Engineering Laboratory (SEL) work. Over several years, this well-known laboratory has conducted several studies and experiments for the purpose of understanding, assessing, and improving software and software processes within a production software development environment at the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) [1].

With our approach we do not intend to create a new software engineering laboratory. Instead, we intend to create a space (virtual or physical) that allows us to conduct empirical studies in the software engineering area by involving students that are enrolled in our current software engineering courses (both at undergraduate and postgraduate university programmes).

Unlike other mature disciplines, the field of software engineering continues to lack a research and development infrastructure that supports systematic testing of novel software engineering methodologies. Our intention is to develop a new experience factory approach based on one explicit educational environment. Initially, we will work just with students as subjects of our first empirical studies. We are fully aware that we will face some problems with the validation of the results that we will be obtained in our student-based experiments. It is impossible to be sure that techniques evaluated under such circumstances will scale up to industrial size systems or very novel software engineering problems. Even though, Kitchenham says that "students are the next generation of software professionals and, so, are relatively close to the population of interest" [3]. In the opposite, students in psychology studies are not representatives of the human population as a whole [4].

In this paper, a description of the state-of-the-art related with the subject of this research is presented in Section 2. Section 3 describes in detail the research objectives and the methodological approach. In Section 4, the past work and preliminary results already done in the context of this research are briefly described. Section 5 presents the future work and expected results for the next 2 years of research. Finally, in Section 6 some conclusions are presented

II. STATE-OF-THE-ART

The state-of-art of this work essentially relates to: ESE (Empirical Software Engineering), SPI (Software Process Improvement) and PM (Project Management). We will give special emphasis to the ESE with students as subjects of experiments.

A. Empirical Software Engineering

ESE is a sub-field of software engineering which aims at applying empirical theories and methods for the measuring, understanding, and improvement of the software development process in real software companies [5]. This definition extends the concept for ESE proposed by Basili, when he said that "experimentation is performed in order to help us better evaluate, predict, understand, control, and improve the software development process and product" [6]. In the early nineties, the empirical methods applied in software engineering were basically restricted to quantitative studies (mostly controlled experiments). The concept of experimental software engineering has moved to empirical software engineering when a range of qualitative methods have been introduced, from observational to ethnographical studies. In a broad sense, an empirical investigation (synonym of empirical study) is a process that aims to discover something unknown or to validate hypotheses that can be transformed in generally valid laws [2].

978-0-7695-4777-0/12 $26.00 © 2012 IEEE
DOI 10.1109/QUATIC.2012.29

It is important to be able to evaluate new techniques and methods in a structured way before they are introduced in the software process [7]. Empirical methods have gained increased attention in software engineering; there are dedicated conferences such as the International Conference on Evaluation and Assessment in Software (EASE), and there are dedicated journals such as the International Journal of Empirical Software Engineering.

Controlled experiments are the most commonly used empirical methods in software engineering. Sjøberg *et al.* define controlled experiment in software engineering as a "randomized experiment or a quasi-experiment in which individuals or teams (the experimental units) conduct one or more software engineering tasks for the sake of comparing different populations, processes, methods, techniques, languages, or tools (the treatments)" [8]. Sjøberg *et al.* analyzed in detail 103 scientific articles published in leading software engineering journals and conferences in the decade from 1993 to 2002 that reported controlled experiments in which individuals or teams performed one or more software engineering tasks.

Currently, some universities offer courses in the ESE area, as in the cases of Norwegian University of Science and Technology [5] and Lund University in Sweden. Both institutions have worked with students as subjects of experiments. These institutions run the experiences out of the courses' context, whereas in our approach the students perform the experiments as part of their regular academic courses. The Department of Computer Science of the University of Helsinki created an experimental software laboratory for basic and applied software development research and education. The name of this laboratory is *Software Factory* and they involve researchers, students, and industry partners in their projects [9].

B. ESE using Students versus Profissionals

In this section, based on literature review we will describe the strengths/weaknesses of using students versus professionals in the empirical software engineering context.

In the survey conducted in [8], a total of 5,488 subjects took part in the 113 experiments investigated, eighty-seven percent were students and nine percent were professionals. This survey demonstrates the importance of using students in this context.

In many studies, students are used instead of professional software developers, although the objective is to draw conclusions valid for professional software developers. The differences are only minor, and it is concluded that software engineering students may be used instead of professional software developers under certain conditions. Höst *et al.* [10] argue that the main reason to use students as subjects is often that they are available at universities and they are willing to participate in studies as part of courses they attend. In many cases, it is possible to combine the learning objectives of the courses with the research objectives of the studies. Tichy refer that software students are much closer to the world of software professionals than psychology students are to the general population [11]. In particular, software graduate students are so close to professional status that the differences are marginal. Software graduate students are technically more up to date than the "average" software developer who may not even have a degree in computing. Software professionals, on the other hand, may be better prepared in the application domain and may have learnt to deal with systems and organizations of larger scale than a student.

Sjøberg *et al.* [12] argue that the main reason of most subjects in software engineering experiments are students is that they are more accessible and easier to organize, and hiring them is generally inexpensive. Consequently, it is easier to run an experiment with students than with professionals and the risks are low. Jaccheri [13] refers that empirical studies are often carried out with students because they are viewed as inexpensive subjects for pilot studies. Svahnberg [14] refers that the students are readily available, often willing to participate, and require no or little compensation. The bad thing is that the variations among studies conducted with professionals are higher than the variations among students due to the more varied educational backgrounds and working experiences in the professionals [12].

Carver *et al.* [15] have developed a checklist that provides guidance for researchers and educators when planning and conducting studies in university courses. In our PhD work, we want to specialize this framework to the software engineering domain, when conducting experiences related with software process improvement and project management research questions.

C. Software Process Improvement

According to Humphrey [16], a software process is "the sequence of steps required to develop or maintain software, aiming at providing the technical and management framework for applying methods, tools, and people to the software task". Therefore, SPI aims at providing software development companies with mechanisms for evaluating their existing processes, identifying possibilities for improving as well as implementing and evaluating the impact of improvements [17].

SPI is an applied academic field, rooted in the software engineering and information systems disciplines, which has been studied for almost twenty years now. It deals primarily with the professional management of software companies, and the improvement of their practice, displaying a managerial focus rather than dealing directly with the techniques that are used to write software. Classical SPI techniques relate to software processes, standardization, software metrics, and process improvement. Many of the major contributions to SPI are originated from the SEI (Software Engineering Institute) at Carnegie Mellon University [18] [36].

SPI is based on process assessment. Most process improvement models and standards applied in SPI primarily provide guidance for process assessment. When critical-mission software is required to demonstrate (often by

obtaining certain type of certifications) their ability to develop and sustain high maturity practices is mandatory. There are currently some software process models available for assessing and improving software development and its related practices.

Empirical studies that we will perform during the PhD work will concentrate primarily on the software development process, from the perspective of process improvement. Thus, we intend to implement experiments involving the suggested practices in CMMI (Capability Maturity Model Integration) [19] and RUP (Rational Unified Process) [20].

D. Project Management Approaches

One of the standard models most popular in PM area is the PMBOK (Project Management Body of Knowledge) [21]. Thus, in 1996, the first version of the body of knowledge in PM was published by the Project Management Institute [22]. According to the PMBOK, projects are composed of processes. A process is "a set of interrelated actions and activities performed to achieve a pre-specified product, result or service. Each process is characterized by its inputs, the tools and techniques that can be applied, and the resulting outputs" [21].

Today, one can find several approaches that aim at collecting PM data in a standardized data model which can be used to implement PM tools and to exchange project data. In order to perform PM activities, people use different methodologies according to their needs and standards. Instead of creating a project plan manually, companies use PM tools that support most important PM processes [21]. For instance, *Microsoft Office Project* is one of the most often used PM tools in small teams [34]. Although it is not based on an official standard, it can surely be considered as a de-facto standard because of its market position. However, this tool does not have an open structure since it uses a proprietary data model, which is not defined by an independent body.

PROMONT [35] is an ontology-based PM approach that intends to summarize all major PM standards and tools in one integrated reference model. It offers extending definitions of PM issues aimed at supporting interoperability of PM systems, processes and organizations. In particular, PROMONT offers a formal approach to define relationships and conditions between different terms that are used in PM.

III. RESEARCH OBJECTIVES AND METHODOLOGICAL APPROACH

A. Research objectives

It is common knowledge that software projects have a high rate of failure [23]. Although various strategies have been tried (such as structured programming, rapid prototyping, CASE tools and so forth), there is still no end to the software crisis.

With the intensification, acceleration in the rate of change, and expansion in the use of information technologies, particular attention is being focused on the opportunities and difficulties associated with sharing knowledge and transferring "best practices" within and across organizations [24]. A best practice is public knowledge, a tactic or method that has been shown through real-life implementation to be successful [25]. Models and standards that provide guidance for process improvement include a set of best practices for product and service development and maintenance [19].

A typical problem with software engineering research is that either it is difficult to find companies that provide reasonable research possibilities or the research is made with students in "artificial environments". Our approach provides a solution for this problem. In our approach we can do research in a very similar authentic environment. The participants in our experiments are students but the environment is very business-like. Teams work constantly together just like in a real work place. There is always a real business demand behind the project, which makes the project context valid for research. Researcher can observe team members anytime and even participate in projects if it is considered useful. Face to what we could allow in real company, our approach has some advantages, namely:

- The ease of research to use their own means of investigation and, at any time, the ease of the researcher to ask participants to answer questionnaires (paper or web) during the semester (within the classes or outside classes);
- All artifacts and documents (e.g. code, models and reports) provided by the teams are available for research purposes (we adopt direct analysis of artifacts to assess the teams process and product maturity);
- Researcher can go to the laboratory and do direct observation (teams have mandatory meetings in our laboratories and are available to be observed when interacting and working in their projects);
- Researcher can take part in the projects and interview both team members and clients during and after the projects.

This PhD thesis will adopt four main objectives. The first three correspond to specific software process research questions that are perfectly pertinent to be addressed when considering the configuration of process frameworks and PM tools in small software development teams. The fourth objective is related with the ESE perspective to assess empirical results with students; which means that efforts relative to this fourth objective must run in parallel with the others. The efforts relative to the first three objectives may not necessarily be run in a sequential order; we will adopt spiral approach to deal with the complexity of managing the complexity relative to all the existing interdependencies between the variables under study in the first three objectives:

- The first objective is to analyze the coverage of CMMI practices that we can expect when adopting the RUP reference model. To fulfill this objective, we need an alignment between CMMI and RUP process frameworks, by selecting the process areas, the specific goals and the specific practices from CMMI and comparing them with the coverage we can expect from the execution of the activities and tasks established by RUP.
- The second objective is to evaluate how CMMI ML2 and ML3 can be accomplished by particular configurations of RUP for small software development teams. To fulfill this

objective, we need to address the specific configurations of RUP and understand the implications in the alignment established in the pursuing of the previous objective. The outcome of these two first objectives may explain how to adopt RUP as a process asset to promote CMMI assessments, taking into account the specific characteristics of the team's organization (roles, tasks, activities).

- The third objective is to assess the impact of PM tools in the performance of software development teams. With this objective we intend to determine the relationship between the maturity of the teams and the support they can get from PM tools. The outcome of this third objective may explain what kind of key success factors we should look for when choosing one PM tool taking into account the process framework (in our case, configurations of RUP for small teams) and the maturity assessment reference model (in our case, CMMI ML2 and ML3) we adopt to frame the software development team.
- Finally, the last and most important objective is to validate the research results to be produced by the previous three objectives in an explicit educational context. The external validity is a major concern in the ESE. The external validity defines the conditions that limit the ability to generalize the results of an experiment to industrial practice. Problems can occur due to the population of participants not be representative of the population under interest, instrumentation is not suitable for industrial practice, and the experiment can be run in a day or special time that will affect the results. In our case, we will run three sets of experiences with students, each one dedicated to one the objectives previously referred. This fourth objective corresponds to an umbrella research question that will enable the production of some systematic insight of the advantages and drawbacks of conducting empirical studies with software students.

B. Methodological approach

An experience should be treated as a process of formulation or verification of a theory. In order that the process provides valid results, it must be properly organized and controlled, or, at least, monitored. In order to achieve these goals several methods of organization of experiments have been proposed. In order to compare the experimentation methodologies we have to consider their different characteristics, for example, the phases of process experimentation, the way of the transformation of abstract concepts of the domain to concrete metrics, the main purpose of experimentation, tools, etc.

In the sub-field ESE, the most relevant research methods are the controlled experiments, the surveys, and the case studies. The selection of methods for a given research project depends on many local contingencies, including available resources, access to subjects, opportunity to control the variables of interest, and, of course, the skills of the researcher [26]. All the research methods have known flaws and each can only provide limited, qualified evidence about the phenomena being studied. However, each method is flawed differently and viable research strategies use multiple methods, chosen in such a way that the weaknesses of each method are addressed by use of complementary methods [27].

We will adopt surveys as one of the research methods (specifically, questionnaires) since it is an assessment tool that can be applied to a considerable number of students, it is cost effective and non-invasive, provide quantitative data, and allows the analysis of results with promptness. It has been argued that the application of questionnaires consumes less time, effort and financial resources than other methods of data collection such as interviews and document reviews [28]. However, at later stages of the research, we will make some interviews with some students to get additional information about the team's organization (mainly related with the instantiation of RUP configurations).

State-of-the-art will be performed as another research approach at initial stages of the PhD work. This activity will complement the brief state-of-the-art presented in this paper. With the literature review, we intend to acquire knowledge about the efforts made for similar problems. We intend to review the following main areas of study:

- Experimental software engineering giving special attention to studies conducted with students as subjects;
- Software process improvement approaches, in particular CMMI and RUP configurations for small teams;
- Project management tools and their support to software development activities.

The three sets of experiences with students will be run as empirical software engineering studies, framed by all the recommendations contained in the previously referred literature. Simultaneously, with the validation of the research results, we will start the development of a framework that shows us a consistent process of using students as subjects of empirical studies. The writing of the thesis will be done along the realization of the work.

IV. PAST WORK AND PRELIMINARY RESULTS

This PhD work takes place within the Software Engineering and Management Group (SEMAG) from the ALGORITMI Research Centre at the University of Minho. SEMAG research group is devoted to study the development process of software-based information systems and related methodologies, focusing on both the engineering and management aspects.

At the undergraduate level (Bologna 1st cycle), the teaching staff of the SEMAG is mainly enrolled in the University of Minho DLic degree in Information Systems and Technology (LTSI) by running, among others, the Software Process and Methodologies (PMS) and Development of Software Applications (DAI) courses. At the postgraduate level (Bologna 2nd cycle), the teaching staff of the SEMAG is enrolled both in the DEng degree in Engineering and Management of Information Systems (MEGSI) and in the MSc degree in Information Systems (MSI) by running, among others, the Analysis and Design of Information Systems (ACSI) and Project Management for Information Systems (GPSI). The empirical studies planned for this PhD work will use software engineering materials and students from PMS, ACSI, DAI, and GPSI courses.

During the first academic semester, PMS students (undergraduation) perform part of the RUP inception phase relative to one real software application, resulting in a project proposal to be addressed to one real client. They have three moments of evaluation and their work focuses on business modeling, requirement, and project management disciplines. The existence of a real costumer permits the acquiring of all the needed information to perform the project proposal. Simultaneously, some ACSI students (postgraduation) get involved with PMS students in order to collect information about the produced business and requirements artifacts and to perform CMMI assessments.

In the second academic semester, DAI students (undergraduation) continue to serve the same client of the first semester and perform the remainder of the RUP inception phase and execute the elaboration, construction and transition phases of RUP to deploy the software application to the real client. Simultaneously, some GPSI students (postgraduation) get involved with DAI students to collect information about the produced software artifacts and the adopted RUP configuration and to perform CMMI assessments and to analyze the utilization of PM tools.

In our approach, we detain several mechanisms that bring into the educational context some characteristics of a real industrial project:

- We have a real client that interacts with the teams and that opens for them the real organizational environment where the software application will be explored;
- We adopt a real problem, with the complexity and the imperfections of any real medium-size software project;
- The inter-relation between PMS and ACSI courses (by means of the ACSI students that emulate external process consultants) and between DAI and GPSI courses (by means of the GPSI students that emulate senior project facilitators) allow us to recreate a typical industrial environment where we have outsourcing of consultants and several depths of professional experiences in the teams;
- The teams compete with each other to sell their software application to the client, which emulates reasonably well the real software market.

The two sets of undergraduate and postgraduate courses (PMS+ACSI and DAI+GPSI) allow us to perform empirical studies of the controlled experiment type, where teams of students (subjects) are the experimental units that lead several software engineering tasks to assess different software processes (RUP configurations) and PM tools support.

In the academic year of 2010/2011, a controlled experiment was performed to assess the reduced model of RUP [29] [30]. It involved seven development software teams. The teams had between 13 and 17 students (1 team with 13, 3 teams with 14, 2 teams with 16 and 1 with 17). Two teams (team 5 and team 7) were randomly chosen to not adopting the RUP reduced model (we called these two teams the "Control Teams"), while the other five teams followed the guidelines established by the RUP reduced model, executing the phases of inception, elaboration and construction. The students elaborated the project proposals during the first semester and developed the software applications during the second semester.

The assessment of the RUP reduced model was conducted by adopting the CMMI-DEV v1.2 ML 2 reference model. With the exception of SAM (Supplier Agreement Management), all the other process areas were assessed. Figure 1 shows the percentage of accomplishment of all specific practices from all process area analyzed for each team. Although there is a significant difference between the various teams, the obtained results show that when the teams use the RUP reduced model they are able to accomplish CMMI ML2 adequately [31].

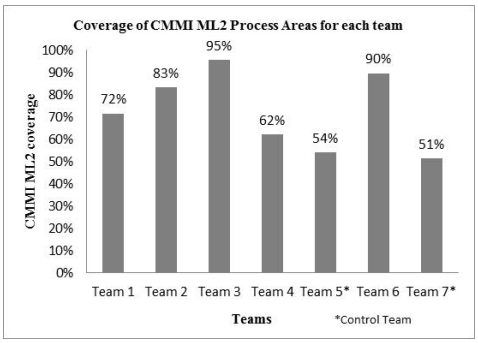

Figure 1. : Coverage of CMMI ML2 Process Areas

In this first experiment, students were suggested to use *Microsoft Project Server 2010* to support their software development activities. The configuration of this platform was performed by two GPSI students. The configuration was extremely difficult to perform. Teams had very little tool support to perform PM tasks.

In the academic year of 2011/2012, a second controlled experiment is being performed to assess the mapping between specific practices of CMMI ML2 and ML3 process areas and RUP artifacts, activities and tasks. In this second experiment, students are using *Clocking IT* [32] and *Teamwork Project Manager* [33] to support their software development activities. *ClockingIT* is an open source application hosted for tracking all tasks, issues, projects and time spent, with a focus on software development and handling large amounts of tasks. *Teamwork Project Manager* is an online application that helps organize and take control of our current projects, task lists, milestones, files, notebooks, resources and time. We intent to assess the influence of these tools in the team's performance. Meanwhile, we are gathering information to elaborate our framework to support the adoption of student teams to perform industry-valuable empirical software engineering experiences.

V. FUTURE WORK AND EXPECTED RESULTS

For the next two academic years (2012/13 and 2013/2014), students will get a more stable PM tool support. With the lessons learned from the two first experiments we intend to refine our processes of experimentation and start to explicitly address specific issues related with conceptual elaboration of our framework. We will also compare the CMMI maturity of teams that adopt the RUP reduced model

with those adopting agile methods. We will also assess specific PM tools.

VI. CONCLUSIONS

Empirical studies in software engineering are important to be conducted to evaluate new tools, techniques, methods and technologies in a structured way before they are introduced in a real software process. Taking into account that: (1) software companies are not usually available to conduct empirical studies; and (2) when, exceptionally, they decide to do it, they keep the results for themselves; empirical studies with students are an interesting alternative to assess software processes and tools and share the results with the academia and the industry.

The problem with this interesting alternative is that there is a lack of scientific evidence that empirical studies with students are valuable for software companies. In our PhD work we intend to develop a framework that shows us a consistent process of using students as subjects of empirical studies. The framework will help to guide new empirical studies in a way that software companies may get interested in buying empirical studies to our laboratory. With this research we hope to contribute to the body of knowledge of ESE, SPI and PM and also to contribute to the increasing of the competitiveness of software companies.

REFERENCES

[1] V. Basili, G. Caldiera, F. McGarry, R. Pajerski, G. Page, and S. Waligora, *The Software Engineering Laboratory-an Operational Software Experience Factory*, in ICSE 92, pp. 370-381, 1992.

[2] A.D. Lucia, F. Ferrucci, G. Tortora, and M. Tucci, *Emerging Methods, Technologies and Process Management in Software Engineering*. John Wiley & Sons, 2008.

[3] B.A. Kitchenham, S.L. Pfleeger, L. M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, and J. Rosenberg, *Preliminary Guidelines for Empirical Research in Software Engineering*, in TSE, vol. 28, no. 8, pp. 721-734, 2002.

[4] R. Rosenthal, *Science and Ethics in Conducting, Analyzing, and Reporting Psychological Research*, in Psychological Sciense, vol. 5, no. 3, pp. 127-134, 1994.

[5] L. Jaccheri and T. Osterlie, *Can We Teach Empirical Software Engineering?*, in METRICS 2005.

[6] V. Basili, R.W. Selby, and D. H. Hutchens, *Experimentation in Software Engineering*, in TSE, vol. 12, no. 7, pp. 733-743, 1986.

[7] M. Höst, *Introducing Empirical Software Engineering Methods in Education*, in SEET 2002, pp. 170-179, 2002.

[8] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.K. Liborg, and A.C. Rekdal, *A Survey of Controlled Experiments in Software Engineering*, in TSE, vol. 31, no. 9, pp. 733-753, 2005.

[9] University of Helsinki. (2012, 2012-5-10). *Software Factory*. Available: http://www.softwarefactory.cc/

[10] M. Höst, B. Regnell, and C. Wohlin, *Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment*, in ESE, vol. 5, no. 3, pp. 201-214, 2000.

[11] W.F. Tichy, *Hints for Reviewing Empirical Work in Software Engineering*, in ESE, vol. 5, no. 4, pp. 309-312, 2000.

[12] D.I.K. Sjøberg, B. Anda, E. Arisholm, T. Dyba, M. Jorgensen, A. Karahasanovic, E. F. Koren, and M. Vokac, *Conducting Realistic Experiments in Software Engineering*, in ISESE 2002.

[13] L. Jaccheri and S. Morasca, *Involving Industry Professionals in Empirical Studies with Students*, in ICESE 2007, Germany, 2007.

[14] M. Svahnberg, A. Aurum, and C. Wohlin, *Using Students as Subjects - An Empirical Evaluation*, in ESEM 2008.

[15] J.C. Carver, L. Jaccheri, S. Morasca, and F. Shull, *A Checklist for Integrating Student Empirical Studies with Research and Teaching Goals*, in ESE, vol. 15, no. 1, pp. 35-59, 2010.

[16] W. S. Humphrey, *A Discipline for Software Engineering*. Addison Wesley, 1995.

[17] W.A. Florac, A.D. Carleton, and J.R. Barnard, *Statistical Process Control: Analyzing Space Shuttle Onboard Software Process*, in IEEE Software, vol. 17, no. 4, pp. 97-106, 2000.

[18] B. Hansen, J. Rose, and G. Tjørnehøj, *Prescription, Description, Reflection: The Shape of the Software Process Improvement Field*, IJIM, vol. 24, no. 6, pp. 457-472, 2004.

[19] SEI, "*CMMI® for Development, Version 1.3*, Software Engineering Institute, CMU/SEI-2010-TR-033, 2010.

[20] P. Kruchten, *The Rational Unified Process - An Introduction*, 3rd Edition. Addison-Wesley, 2003.

[21] PMI, *A Guide to the Project Management Body of Knowledge*, Fourth Edition, Project Management Institute, 2008.

[22] PMI. Available: http://www.pmi.org

[23] The Standish Group. (2009). *Chaos Report*. Available: http://www1.standishgroup.com/newsroom/chaos_2009.php

[24] W.J. Orlikowski, *Knowing in Practice: Enacting a Collective Capability in Distributed Organizing*, Organization Sciense, vol. 13, no. 3, 249-273, 2002.

[25] R. G. Cooper, *Winning at New Products: Accelerating the Process from Idea to Launch*, third edition, Addison-Wesley, 2001.

[26] S. Easterbrook, J. Singer, M.A. Storey, and D. Damian, *Selecting Empirical Methods for Software Engineering Research*, in Guide to Advanced Empirical Software Engineering, 1st Ed., pp. 285-311, 2008.

[27] J. W. Creswell, *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*, 3rd edition,Sage Publications Inc., 2009.

[28] I. Garcia, C. Pacheco, and P. Sumano, *Use of Questionnaire-Based Appraisal to Improve the Software Acquisition Process in Small and Medium Enterprises*, in SERMA, vol. 150, pp. 15-27, 2008.

[29] P. Borges, P. Monteiro, and R. J. Machado, *Tailoring RUP to Small Software Development Teams*, in SEAA 2011, pp. 306-309, 2011.

[30] P. Borges, P. Monteiro, and R. J. Machado, *Mapping RUP Roles to Small Software Development Teams*, in SWQD 2011, pp. 59-70, 2012.

[31] F. Mandjam, *Avaliação do Impacto das Práticas do CMMI no Desempenho de Equipas de Desenvolvimento de Software no Ensino*, MSc in Engineering and Mangement of Information Systems, Universidade do Minho, Portugal, 2011.

[32] E. Simonsen and E. Simonsen. (2008, 2012-05-11). *Clocking IT TimeTracking 2.0*. Available: http://www.clockingit.com/

[33] Teamwork Project Manager. (2012, 2012-05-11). *Teamwork Project Manager*. Available: http://www.teamworkpm.net/

[34] Microsoft. (2012, 2012-05-10). Available: http://office.microsoft.com/pt-pt/project-help/ CH010362755.aspx

[35] S. Abels, F. Ahlemann, A. Hahn, K. Hausmann, and J. Strickmann, *PROMONT - A project management ontology as a reference for virtual project organizations*, in LNCS, vol. 4277, pp. 813-823, 2006.

[36] M.C. Paulk, "*A History of the Capability Maturity Model for Software*," ASQ Software Quality Professional, vol. 12, no. 1, pp. 5-19, 2009.