ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2005, Vol. 34, No. 1

SOFTWARE ENGINEERING PROCESS AND ITS IMPROVEMENT IN THE ACADEMY[1]

Eduardas Bareiša, Eimutis Karčiauskas, Virginija Limanauskienė, Romas Marcinkevičius, Kęstutis Motiejūnas

Software Engineering Department, Kaunas University of Technology
Studentų 50-406, LT-51368 Kaunas, Lithuania

Abstract. Changes in software technology and models for software development require commensurate change in the education of software developers. One way of teaching software engineering is to organize a course around a project similar to a real industrial project. The educational community itself is increasingly moving from lecture-format courses to team projects, problem-solving, direct involvement with actual development, and other formats that require students to exercise the ideas they are learning. In this paper we investigate the software engineering process improvement at Kaunas University of Technology to let students experience the realistic software engineering problems and environments during their software engineering education. Organizing the software engineering process we implemented some key practices of repeatable level of Capability Maturity Model.

1. Indroduction

Software Engineering (SE) is not simply concerned with technical activities, but it also involves both managerial aspects, e.g., definition, organization and planning of multiple interrelated activities, and collaborative aspects, e.g., information exchange and coordination of people working to fulfil a common objective.

Changes in software technology and models for software development require commensurate change in the education of software developers. First, the educational institutions themselves must be able to adapt quickly, both in the content of their offerings and in their ability to exploit new technology in support of education. Second, the educational institutions must prepare their graduates to assume responsibility for upgrading their own skills throughout their careers [1].

Students are in general skilled in programming and up-to-date in current technology innovations, but have a poor SE background. I.e., they have no experience in applying known SE concepts and methods to practical development problems. Also, they are not used to teamwork, i.e., students often work individually and dislike collaboration, and mutual responsibilities and they are not used to communicate and present fruitfully their work, or to discuss with people with a different background [2].

Apart of the needed technical preparation, the SE market requires graduated people to be dynamic, collaborative, and professionally independent. Companies in the field often observe that graduated employees need first a training period to learn how to put into work their professional skills in real projects.

SE undergoes continuous evolution, because it is a relatively young discipline and because it strongly depends on evolution in both current technology and methods. For these reasons, SE study programs do not adhere to any common structure, but they rather try to answer requirements of the local market's reality. Hence, both theoretical and practical aspects tackled by the programs are diverse. Many educators choose projects as the educational tool to meet SE requirements of both industrial and academic sides. For instance, the SE course at Politecnico di Torino [3] is based on a project carried out in collaboration with an industrial organization playing the role of the customer, and a general process model formalized in a quality manual given to the students. The overall objective of this paper is investigating of the SE process improvement at Kaunas University of Technology (KUT) to let students experience the realistic software engineering problems and environments during their SE education.

[1] This work was supported by Lithuanian State Science and Studies Foundation, award B-06/2003

2. Background. SE study program at KUT

The aim of Software Engineering Master Studies is to prepare software development leaders – the future chief engineers, project executives, experts that can both acquire and implement efficient design methods and new technologies in practice. The study program provides the possibilities for preparing of professional development of software systems and for scientific research. Both in the world and in Lithuania the demand for software projects executives of high quailfication is large. Software Engineering Masters will be able to work in various Lithuanian and foreign information technology and telecommunication enterprises.

2.1. Program reasoning and relevance

In the first NATO Software conference that was held three decades ago Software Engineering was defined as a discipline and a profession. However, the ultimate formalization of this profession was performed only in the last decade and many universities incorporated Software Engineering studies in their study plans.

Currently Software Engineering knowledge areas are defined and accreditation regulations of specialist education programs are discussed globally. In regularizing Software Engineering knowledge areas, the study content and programs the main task was to separate them from the appropriate computer science and computer engineering areas and study programs [4].

The main work in defining Software Engineering Body of Knowledge was done by SWECC (Software Engineering Coordinating Committee). The second stage (Stone Man) of Software Engineering Studies program project was presented in 2000 [5]. Software Engineering Bachelor and Master Programs of 27 universities from various countries were analyzed in the first stage of the project (Straw Man). It was found out that only four universities prepare the Bachelors of this area and the rest universities prepare Masters. Software Engineering Master Study Program in the Informatics Faculty of KUT was created under the recommendations of SWEBOK.

2.2. Study program's objectives

The objectives of the Software Engineering Master Study Program are:

1. To combine theoretical knowledge and practical skills in order that students would be able to design and produce efficiently software systems which fulfil users' and clients' requirements.
2. To provide students with knowledge and experience in such knowledge areas: Engineering Economy, Software Requirements, Software System Design, Testing, Software System Maintenance, Software Engineering Management, Software Engineering Processes, Software Engineering Tools and Methods, Software System Quality.
3. To teach to evaluate, analyze and simulate software system quality factors in order to ensure a disciplined and controllable development of a software system.
4. To teach to choose tools, methods and design methodology applicable to the development environment.
5. To provide with the experience of organizing individual and team work and with skills of communicating and collaborating professionally.
6. To teach to understand and to be able to improve the software engineering process.
7. To develop the understanding of product quality, price, schedule abidance importance in software system production.
8. To teach to prepare technical documentation completely and consistently, to present technical concepts in both written documents and oral presentations.

2.3. Study program

Software Engineering Master studies last 2 years (4 semesters). The program consists of obligatory and elective modules, research and Master's thesis (that together make 80 credits). The overall scope of theoretical study modules consists of 44 credits (55% of the whole study program); the scope of modules for research and practical design together with the Master's thesis consists of 36 credits (45% of the whole study program).

The continuation of Bachelor's study program, the readiness for the research and design is declared in Master's study program. The study program corresponds to the latest Software Engineering development trends; its structure covers all stages of software design process. One way of teaching software engineering is to organize a course around a project similar to a real industrial project [2]. The educational community itself is increasingly moving from lecture-format courses to team projects, problem-solving, direct involvement with actual development, and other formats that require students to exercise the ideas they are learning [1]. The development of the real industrial software systems and practice are designated for the skills nurture of the students. Project lasts 3 semesters and is implemented using 4 research modules: "Requirements Specification", "Software System Architecture Analysis", "Information Technologies in Design Management" and "Software Implementation Research" (practice) (Figure 1). Master students have to apply knowledge acquired in theoretical module studies in practice and with creativeness, therefore one can say that practical design "feeds" on the resources of theoretical module studies. The project is a pivot of the SE study program at KUT. During these three semesters Master students have to develop the software system and implement it at the client's enterprise.

The first semester. In the first semester the main attention is paid to requirements analysis and specification (modules "Software Requirements Analysis" and "Simulation and Validation of Systems"). In such a way Master students learn methods and tools of the requirement analysis already in the beginning of the study program and afterwards they can use the acquired knowledge to form the requirements specification of the objective area under computerization according to the Master thesis. Two courses that generalize software engineering area – "Software Engineering Process" and "Software Engineering Management" – are delivered together with the above mentioned courses; these courses are purposed for expanding and deepening knowledge acquired in Bachelor's studies. During project development students familiarize with the standards, prepare proposal, plan and requirement specification of the project.

The second semester. During the second semester the main attention is paid to the software system architecture design stage. Master students deepen their knowledge in architectural and detailed design of software systems (the module "Software Design"); software design strategy and methods are introduced in this module. Models of the internet and the intranet, documentary and structural objects data models, data model formation according to the organization model are presented in the module "Data Design". Software verification and validation methods, functional and structural testing methods are analyzed in the module "Software Testing Methods". During project development students prepare system and detailed architecture.

The third semester. Only one module which does not depend to the research ones is in the third semester; this module is "Software Development Tools". CASE tools and their usages throughout the whole software life-cycle are analyzed in this module. Practice significance has to be highlighted. During the practice the research is performed in a real enterprise under the guidance of the department and the representative of the enterprise; an additional material for the Master thesis is obtained here. The aims of the practice are to deepen software engineering knowledge, to analyze the management of the software design enterprise and of the projects, to familiarize with design technologies used in the enterprise, the applied standards, business environment, the methodology of negotiation with clients, and to acquire skills of working in the organization. While Lithuania integrates into the EU, it is important to prepare professionals properly in order that our enterprises would be able to compete and to collaborate with the enterprises from other countries. Enterprises wish professionals with experience, especially with international experience. The only way for university graduates to acquire experience is practice. However it is difficult to find Lithuanian companies designing high quality software which would welcome the trainees. Consequently the possibility to practice abroad is very important.

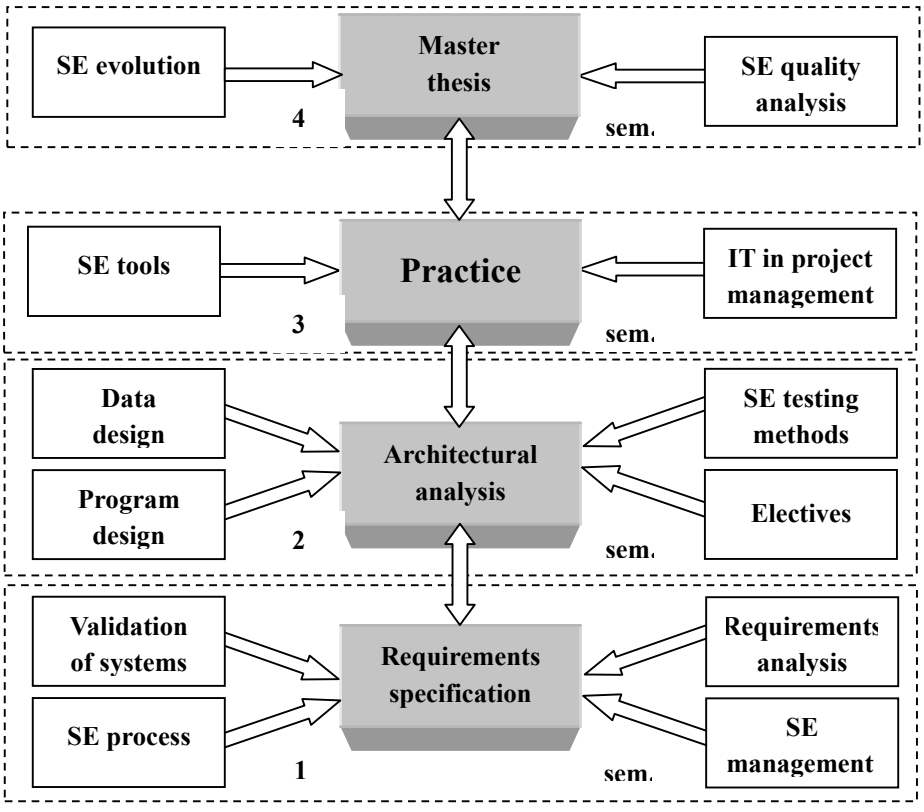

Figure 1. Project as pivot of the SE study program at KUT

The fourth semester. During the fourth semester the main attention is paid to the Master thesis. Two modules are delivered; they complete software life-cycle analysis. Modules are "Software Evolution" and "Software Quality Analysis". One of the fundamental Master thesis sections is purposed for analyzing the quality of software designed during the 1-3 semesters, for its experimental research and refinement, so the material presented in the above mentioned modules is an important support for preparing of Master thesis.

3. The realization of teaching by designing (an educational model)

The main principle of study system is the wholeness of science and studies and design skills training, based on systematic and autonomous work of a student. The importance of practical design is especially highlighted. Master project themes and practice places are chosen according to competence and qualification of students; in order to evaluate this Master students qualification evaluation survey is performed at the beginning of the first semester. In order that students would apply and deepen theoretical knowledge of general matters in real design, they design a large software system and their individual research schedules and deliverables are coordinated. In each stage of design within the given time students create products that have to fulfil requirements set in advance. Students may use either document templates from the client's enterprise or from the Master design study program (in the MS Word, MS Project or HTML formats). The purpose is to train methodical work skills in a real industrial organization. Therefore the „penalties" system is used for the deviation from the project schedule.

E. learning is widely used in studies. Studies management information is public and available on the Internet:

- All the information about the modules is placed in the university study modules database (http:// www. ktu.lt) and is available for the Master students without restraints.
- The learning material is placed by lecturers in their Internet sites.
- Information on research coordination is published in the Internet site http://www.soften.ktu.lt/~virga/mag_atmintine. Project themes, document templates, examples, standards, requirements for deliverables, schedules, requirements for skills, project and software quality requirements are placed there.
- Project theme suggestions are published in the Internet.
- For the management of personal research documents Master students create their project information systems where intermediate products and documents of the project are placed. Such information management systems are private; they can be accessed only with passwords as this is required by the majority project customers.
- Information on software process improvement is published in the Internet site http://proin.ktu.lt /pkp/pkbm/index2.php.
- Hot Master studies' problems are discussed in the forum http://proin.ktu.lt
- Lecturers communicate with Master students through the e-mail, in seminars and individual consultations.

In the end of each semester Master students prepare a report on the work that has been done and present it in the seminar.

4. Evaluation of the experience of teaching by designing

Evaluation should involve an assessment of the product produced, and an analysis of the effectiveness of the process used to create the product. The aim of evaluation is two-fold: to recognise those strategies and techniques which proved effective during the course of the project so that their use may be reinforced or expanded; and to identify areas in the process and product that need to be improved in the next project [6].

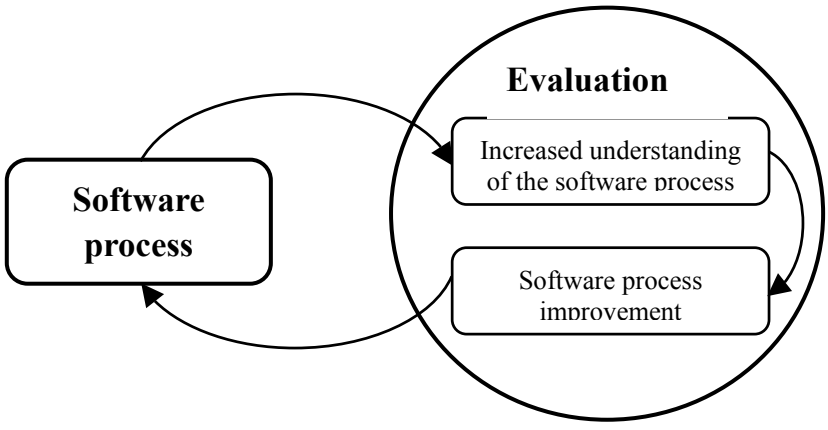

Figure 2. The link between the software process and the evaluation process

The benefit of evaluating a software project is greater understanding of the software project organization, software development organization and people that are involved. This understanding is essential for the ability to perform software process improvements.

The link between the software process and the evaluation is illustrated in Figure 2. This is a rather simplified illustration; the actual evaluation should be seen as a part in the software process. It is not enough to understand the process and then to improve it; goals need to be set. These goals have to be defined before the evaluation; otherwise, the collection of information can get out of hand. Furthermore, unnecessary information will be collected if goals are not defined [6].

After three years of studies according to the Software Engineering study program and having, alas, almost no feedback with manufacturing and scientific research institutions it is difficult to draw wider conclusions about the merits and demerits of this education model. On purpose to fill this gap at least in part after the completing of Master thesis the students and practice tutors answer an anonymous questionnaire. The results of teaching by designing and the assessments are presented below.

Master project. Selecting projects the priority is to such ones that come from manufacturing. Master Project topics corresponded to occupational interests of 90% students. Master students admitted in 2001 and 2002 developed software systems ordered by such firms: "Baltic Software Solutions", Join-Stock Company Elsis, LKSoft Baltic, LKSoft Baltic Kaunas, Join-Stock Company Singleton Labs, "IBS Baltic", the Institute of Cardiology, Join-Stock Company"Virtuali tikrovė" (Virtual Reality), the enterprise of T. Valiukėnas, Lithuanian Naval Forces, Join-Stock Company "Vičiūnai", the Institute of Information Technologies, the National Institute of Forestry, Join-Stock Company "Failanas", H&S Qualita' nel software S.p.A (Italy), Navita srl (Italy).

Assessments:

- The average grade for the project quality stated by Master project tutors is 9.72;
- Master students evaluated their projects on the average 8.33;
- The average grade for the Master project tutors' consultations stated by students is 8.28;
- The average grade for Master project design process arrangement stated by students is 7.71;
- 43% of students referred the number of Master project deliverables as too large, 57% - as sufficient and no student referred this number as too small.

Practice. New relations were settled with the enterprises from Italy, Norway and Germany; some students practiced there. All the executives of both Lithuanian and foreign enterprises evaluated the students' qualification, self-discipline and personal features positively. However during the practice not all the Master students performed the tasks that correspond to the practice goals. It has to be mentioned that the number of enterprises involved in large software systems design in Lithuania is low.

Assessments:

- The average grade for the Master students' work in the enterprises stated by practice tutors is very high – 9.93;
- Only 76% of students consider the practice as useful for their skills development;
- The practice place corresponded to the Master project topic for 62% of students.

5. SE process improvement

Software quality and software process improvement are central topics in modern IT industry. However, there is no standard consensus about how to educate future software engineers in such topics. Software quality and software process improvement comprise both technical and managerial issues. Among the technical, we list design, testing, inspection, and configuration management. On the other hand, the quality and improvement models, like Capability Maturity Model (CMM) [7], ISO 1554 (SPICE) [8], Bootstrap [9], etc. derive from managerial and organisational theories. When teaching technical methods, such as design, one can employ the same educational methods that are commonly used to teach programming languages or mathematics. First, the teacher explains the method and provides examples. Then, the students are asked to solve toy problems, alone or in groups, by employing the method. This educational method does not properly work when applied to teaching of the managerial part of software process improvement and software quality [10].

Concerning the SE process organization, the use of a defined process model is an important step forward as it permits the students to better understand the work to be done, and thus better focus on engineering activities [2]. The process model followed by the students at KUT is a waterfall process model. The process model is organized into three main development phases. The requirement engineering phase focuses on use case driven requirement engineering by refining the initial high-level use case specification. The construction phase consists of both object-oriented modelling and implementation, whereas the testing phase focuses on both unit and integration testing, and produces documentation of test specifications and results. All phases produce documentation of the corresponding results.

Software process improvement concepts, applied in an effort to increase productivity and quality in industry are also needed in the academic environment. By applying process improvement we can prepare students for the future and simultaneously improve our own understanding and teaching of the software engineering. The reasoning is, if attention to software

process improves the commercial software development environment, then the application of software process techniques should also strengthen the classroom environment. Two positive effects could be expected. First, successful experience with the techniques on the class project would result in students even better prepared to meet the challenges of modern software technology. Second, the use of quality improvement techniques, applied to the software engineering project, would be a step in the direction of improving academic education [11].

Software process issues become even more important in the classroom, working with inexperienced students. Using the CMM [7] role definitions and job descriptions in classes can ease students transition to working as a cohesive software engineering team in a professional environment. They have a better appreciation of why software development can be so difficult and it gives students a high-level model for reducing this complexity. The transfer of knowledge from university to industry should begin to include process material in software engineering classes. Even if they do not apply most of the ideas on a class project, students can gain a clearer idea of the complications inherent in developing large software products and how they can be managed [11].

The most common way of transforming a student project into a real industrial experience is the involvement of an industrial organization to play the role of the customer, see [6, 12]. This is a valuable for ensuring that the project be a successful educational experience for the students who feel in fact strongly motivated. However, this choice requires a lot of work from both academic and industrial sides. Other ways of transforming a student project into a real industrial experience is to adopt industrial CASE tools, industrial standards and/or a process model from an industrial organization [2, 13].

The Capability Maturity Model developed by the Software Engineering Institute provides a theoretical framework for the software engineering process improvement. This model consists of five levels of maturity and the key practices that organizations must implement to achieve each level. Table 1 provides a definition of the various levels of sophistication and allows the measurement of one organisation against another [7].

Table 1. Maturity levels defined in Capability Maturity Model

Level	Description
1. Initial	Software process is ad hoc and occasionally chaotic. Few processes are defined and success depends upon individual effort and heroics.
2. Repeatable	Basic project management processes are established to track cost, schedule and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3. Defined	The software process for management and development activities is documented, standardised and integrated into the organisation. All projects use an approved, tailored version of the process.
4. Managed	Detailed metrics of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5. Optimised	Continuous process improvement is enabled by the quantitative feedback from the process and from piloting innovative ideas and technologies.

As a rough guide, most software development organisations would like to be at, or would be satisfied with level 3; fewer reach level 5. However, based on observations most organisations have a process flip-flop between levels 2 and 1. Those with no process at all would undoubtedly remain at level 1, with the emphasis on individual effort and heroics for any degree of project success [14].

Of course, the maturity of software engineering process at all or almost at all universities is at initial level. Organizations at the initial level are assumed to have a chaotic software engineering process and they would need to implement some key management practices: Requirement Management; Software Project Planning; Software Project Tracking and Oversight; Software Subcontract Management; Software Quality Assurance; and Software Configuration Management. These management practices help to put in place a basic disciplined process with the aim of obtaining a repeatable process.

Improved process maturity results in an increased productivity, better quality and more accurate schedule time. However, it is difficult for most organizations to even achieve the first step of the ladder (Level 2), since there is not a practical implementation strategy for the key practices within each level. There is a need for determining an implementation strategy for the key practices within each level according to the support they provide each other [15].

On the other hand, small software organizations, and small team projects (such are all students projects at universities) may find it difficult to achieve higher levels of maturity according to the CMM, since many of the key practices suggested by this model are inappropriate to small businesses and projects. Small project teams cannot cope with the overheads produced by the amount of documentation required by

the CMM and they must use combined documents to reduce time. In small projects, teams usually have a flat structure, resulting in developers being assigned several roles due to scarce resources. This contrasts with the team structure and positions suggested by the CMM practices and makes the implementation of some practices difficult [15].

Organizing the software engineering process at KUT we strive to implement some key practices of repeatable level, that are appropriate for student projects. Below we will shortly summarize the main results of software engineering process improving at KUT.

Requirements management. The requirements elicitation phase is standardized. To achieve this, we have created a requirements analysis template with standard sections. The requirements document is subsequently used for controlling the requirements changes or/and the introduction of new requirements at any time and for the acceptance of the final product by the client and by the teacher. Different versions of the requirements document are kept and each of these versions includes the requirements changes made to the previous one.

Project planning and tracking. The project planning and tracking process is supported by Microsoft Project tool. The students create an initial plan, which subsequently is continually updated and refined. The Microsoft Project allows us to: keep track of actual versus estimated; produce status report at any time, earlier recognize risks.

Subcontract management. There is no subcontract management at KUT and there are no plans to implement it. The reason is very simple – the student's projects are too small.

Configuration management. Now there is no configuration management at KUT. The reason is the same– the student's projects are too small. However, we plan to introduce bigger projects with teams involving more than 5 students. In this case some configuration management practices will be implemented.

Software quality assurance. We have created design, detailed design and test plan templates with standard sections. The design and development process has been aided by the usage UML tools. The usage of UML tools helps us to standardize the development process. The unit testing is standardized too. The test cases in the system test plan are aligned with functional requirements.

6. Conclusions

The advantages of teaching by designing are:

- Design corresponds to the latest Software Engineering development trends;
- Design is supported by knowledge acquired in studies of theoretical modules;
- Design process corresponds to the software life-cycle;
- The relation with manufacturing is realized through practice;
- Software engineering process includes some key practices of repeatable level of CMM model.
- The practice of Master students in foreign software design companies allows the staff of the Informatics faculty gaining experience from the Western Europe enterprises, being aware of contemporary requirements for professionals;
- IT tools widely used in education for the collaboration and the communication develop contemporary work organization skills;
- The tool of e. learning – Master studies site on the Internet allows presenting the latest information, coordinating the design, having a feedback with the students.

Problems that have to be solved:

- A week feedback with manufacturing and research institutions;
- Not all the Master students practise in manufacturing enterprises;
- The internet system for the management of Master project design has to be created.

The objective of education is learning. Even in the classroom, the objective of teaching is to create a fertile setting for the student to learn. After graduation, though, the student becomes responsible for his or her own further education. Even with the best graduate education, software developers – especially software engineers – will need to periodically update their skills and their mastery of new technology. So one of the responsibilities of the formal education is to prepare the student with skills for independent lifelong learning.

References

[1] **M. Shaw.** Software engineering education: A roadmap. In The Future of Software Engineering., *New York, ACM*, 2000, 371–380.

[2] **M.L. Jaccheri, P. Lago.** How Project-based Courses face the Challenge of educating Software Engineers. *Proc. of the joint World Multi-conference on Systemic, Cybernetics and Informatics* (SCI'98) *and the 4th International Conference on Information Systems Analysis and Synthesis* (ISAS'98), *Orlando, USA, Jul*. 1998.

[3] **M.L. Jaccheri, P. Lago.** Applying Software Process Modeling and Improvement in Academic Setting. *Proceedings of the* 10*th Conference on Software Engineering Education & Training, Virginia Beach, Virginia, IEEE Computer Society Press, Apr*. 1997.

[4] Computing Curricula 2001. Internet access: *www.computer.org/education/cc2001/final/, last visited* 2004 11 22.

[5] Guide to the Software Engineering Body of Knowledge. *Internet access: http://www.swebok.org, last visited* 2004 11 15.

[6] **J.M. Hogan, G. Smith, R. Thomas.** The Real World Software Process. *Software Engineering Conference* 2002, ISSN: 1530-1362, *Ninth Asia-Pacific Publication*, 366- 375

[7] Software Engineering Institute, Carnegie Mellon University. *CMMI for Systems Engineering. Software Engineering, Integrated Product and Process Development and SupplierSourcing* (*CMMI-SE/SW/IPPD /SS*) *Version* 1.1, 2002.

[8] ISO/IEC TR 15504. *International Standard for Software Process Assessment*, 2003.

[9] **P. Kuvaja, J. Simila, L. Krzanik, A.Bicego, G. Koch, S. Saukkonen.** Software process assessment and improvement, the BOOTSTRAP approach. *Blackwell Publishers, Oxford, UK*, 1994.

[10] **T. Dingsyr, M. Letizia Jaccheri, A. I. Wang.** Teaching software process improvement through a case study. *Computer Applications in Engineering Education*, *Vol*.8, 2000, pp. 229-234.

[11] **L. Werth.** An Adventure in Software Process Improvement. *In J. L.Dìaz-Herrera (ed.), Software Engineering Education*: 7*th SEI CSEE Conference. New York: Springer-Verlag*, 1994, 191-210.

[12] **H. Saiedian, D. J. Bagert, N. R. Mead.** Software Engineering Programs: Dispelling the Myths and Misconceptions. *IEEE Software, September-October* 2002. *IEEE Computer Society*, 2002.

[13] **B. Meyer.** Software Engineering in the Academy. *In Computer* (*IEEE*), *Vol*.34, *No*.5, *May* 2001, 28-35.

[14] **G. Stone.** Making the most of the software development process. *In Software World*, 2003.

[15] **S. Otoya, N. Cerpa.** An Experience: A Small Software Company Attempting to Improve its Process. *Proc. Software Technology and Engineering Practice, STEP* '99, 1999, 153-160.

DOI: 10.5755/j01.itc.34.1.11974