Overcoming Challenges in Software Engineering Education:

Delivering Non-Technical Knowledge and Skills

Liguo Yu
Indiana University South Bend, USA

A volume in the Advances in Higher Education and Professional Development (AHEPD) Book Series

ENGINEERING SCIENCE REFERENCE
An Imprint of IGI Global

Managing Director:	Lindsay Johnston
Production Editor:	Jennifer Yoder
Development Editor:	Vince D'Imperio
Acquisitions Editor:	Kayla Wolfe
Typesetter:	John Crodian
Cover Design:	Jason Mull

Published in the United States of America by
Engineering Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Overcoming challenges in software engineering education : delivering non-technical knowledge and skills / Liguo Yu, editor.
pages cm
Summary: "This book combines recent advances and best practices to improve the curriculum of software engineering education, bridging the gap between industry expectations and what academia can provide in software engineering education"-- Provided by publisher.
Includes bibliographical references and index. ISBN 978-1-4666-5800-4 (hardcover) -- ISBN 978-1-4666-5801-1 (ebook) -- ISBN 978-1-4666-5803-5 (print & perpetual access) 1. Software engineering--Study and teaching (Higher) I. Yu, Liguo, 1969-
QA76.758.O85 2014
005.107--dc23
2014000885

This book is published in the IGI Global book series Advances in Higher Education and Professional Development (AHEPD) (ISSN: 2327-6983; eISSN: 2327-6991)

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 12
Project-Based Learning:
An Environment to Prepare IT Students for an Industry Career

Luís M. Alves
Instituto Politécnico de Bragança, Portugal

Pedro Ribeiro
Universidade do Minho, Portugal

Ricardo J. Machado
Universidade do Minho, Portugal

ABSTRACT

The lack of preparation of Software Engineering (SE) graduates for a professional career is a common complaint raised by industry practitioners. One approach to solving, or at least mitigating, this problem is the adoption of the Project-Based Learning (PBL) training methodology. Additionally, the involvement of students in real industrial projects, incorporated as a part of the formal curriculum, is a well-accepted means for preparing students for their professional careers. The authors involve students from BSc, MSc, and PhD degrees in Computing in developing a software project required by a real client. This chapter explains the educational approach to training students for industry by involving them with real clients within the development of software projects. The educational approach is mainly based on PBL principles. With the approach, the teaching staff is responsible for creating an environment that enhances communications, teamwork, management, and engineering skills in the students involved.

INTRODUCTION

During Software Engineering (SE) training, it is very difficult to provide industry-standard knowledge and skills, especially non-technical knowledge. These skills can be grouped into three main areas: management, engineering and personal. One challenge facing SE education is that the current lecture-based curriculum hardly engages students. Students often view SE principles as mere academic concepts, which are less interesting and less valuable. The reality is that Computer Science (CS) and Information Systems (IS) graduates often have to develop SE knowledge

DOI: 10.4018/978-1-4666-5800-4.ch012

and skills, especially non-technical knowledge and skills, later on, when they start their careers in industry.

The lack of preparation of SE graduates for a professional career is a common complaint raised by industry practitioners (Karunasekera & Bedse, 2007). One approach to solving, or at least mitigating this problem, is the adoption of the Project Based Learning (PBL) (Barrows & Tamblyn, 1980) training methodology. Involving students in industry projects during their undergraduate degree is a well-accepted method of preparing students for their professional careers.

In our approach we involve students from BSc, MSc and PhD degrees in Computing from our university to develop a complete software project requested by a real client. At the BSc level (Bologna 1st cycle) we involve students from Software Process and Methodologies (PMS) and Development of Software Applications (DAI) courses. At the MSc level (Bologna 2nd cycle) we involve students from Analysis and Design of Information Systems (ACSI) and Project Management of Information Systems (GPSI) courses. The curriculum integration and the pedagogical cooperation, through an integrated project between the four courses in analysis, are intended to promote students to work in a software development environment that is similar to an organization environment. Parts of the contents of these courses were also framed several times in training given by the teachers in business or industrial contexts, under protocols between the university and relevant organizations.

In this group of courses we must highlight the DAI course because of the unifying role it plays when compared to the remaining three. DAI has a learning value of 10 ECTS (European Credit Transfer and Accumulation System) and teachers of subsequent courses "expect" from students an effective ability to develop IT (Information Technology) solutions to problems with medium complexity. This main goal drives the teaching team to adopt a set of procedures and pedagogical practices capable of dealing with the complexity inherent in managing a course of this kind.

To perform these software projects, students pursuing the same degree constitute the teams. However, they have to work in close collaboration with teams from other degrees. The teaching staff is responsible for creating an environment that enhances communications skills, team working skills, management skills and engineering skills of the students involved. It is a well-accepted fact that a competent software engineer requires a wide variety of skills in areas such as management, engineering, team working and communication (Ali, 2006; Nunan, 1999).

Another challenge is to evaluate teams and individuals who develop unique industry projects (Clark, 2005). In our case, we use "*Assessment Milestones*" distributed throughout the semester that allow us to track the students' work progress and thus avoid an end-point evaluation only.

Our approach presents an advance in SE education, in order to overcome the aforementioned challenges. The teams have the opportunity to interact with a real client. They can learn and apply SE principles through a real software project. Thus, they can evolve and improve their technical and non-technical skills. In our setting we promote a win-win approach for all stakeholders: clients, students, teachers and researchers. Clients will have state of the art projects implemented in their companies. Students can acquire technical and non-technical skills and work closely with real-world problems. Teachers will have the opportunity to teach technical knowledge authentically and realize new problems that companies are facing. Researchers can perform experiments on new and/or existing techniques, tools and methods. This gives us the opportunity to provide guidelines for SE educators in order to improve their curricula and provide CS students with ready-to-apply SE knowledge skills.

This chapter is structured as follows: Section 1 introduces the range of problems that promoted the present work. Section 2 addresses, in detail, the current challenges in the software PBL that we are trying to respond to, as well as affording a comprehensive state-of-the-art picture that will suit the purpose of validating the strength of our approach. This section will also be dedicated to motivating the reader towards the systematic use of software PBL, which will be the subject of the chapter and that is, essentially, revealing its significance. Section 3 contains our PBL approach. In this section we will explain the environment and the integration of the different teams from all the courses involved. Section 4 shows the main results obtained from our PBL approach. We will start this section with an evaluation of the students teams method used in our approach, followed by a detailed analysis of the work projects developed by ACSI and GPSI students. It will also focus on discussing the results we are going to present in the chapter. Finally, in section 5 we will present our major concluding remarks.

BACKGROUND

SE has brought to the CS field the confluence of the process and development methodologies with the economic surroundings that are found to be indispensable to the professionals working in the industry with roles and responsibilities that go beyond the mere computer programming (Engle, 1989). In SE we find a clear concern with what is beyond purely technical issues, which have always been the ones that truly the CS devoted full attention to. In this sense, it is important for a software engineer to be educated in communication and management skills. These are skills which are vital to the software engineer's success and they cannot be left to be learned by "osmosis" (Engle, 1989).

While SE (as a discipline) is dedicated to study the software process development, IS (as a discipline) analyzes the impact of software-based systems on individuals, organization and society. However, in the scientific context there is a great divide between methods and research questions (Finkelstein, 2011; Gregg, Kulkarni, & Vinzé, 2001). The cross-fertilization between the two disciplines has allowed the cooperative and coordinated development of the engineering and requirement management, modeling and systems architecture, process development and project management approaches (Avison & Wilson, 2001; Birk et al., 2003; McBride, 2003). This is why, in the context of project technology-based solutions in problems of organizational nature, the two disciplines (and corresponding professional performances) merge into a blend of common methodologies, techniques and tools (Fung, Tam, Ip, & Lau, 2002; Hellens, 1997; Jayaratna & Sommerville, 1998) demonstrating the existence of an SE/IS convergence. The understanding of the intersection between the two disciplines allows students to develop projects of better quality, since the whole project is grounded in the knowledge of both disciplines.

When we are to deliver knowledge and skills to IT students we should be able to teach engineering and software management. The success of a software engineer is related to the ability of engineering and software management methodologies to adapt to the huge demands that professionals are subject to nowadays, which include dealing with all procedural issues of software production, along with technological competence and sensitivity to the needs and expectations of users (Platt, 2011).

The rationalization of all decisions relating to issues not explicitly technological is fundamental for a correct (effective and efficient) operation of a software development team (Dutoit, McCall, Mistrik, & Paech, 2006). Into this set of concerns and attitudes also fit the aforementioned software economics issues (Tockey, 1997). When properly reconciled with the technological dimension, *Software Engineering Management* has adopted the designation of *Software Engineering and Management* (Shere, 1988). This designation, which reinforces the existence of a management dimension within the SE (as in any other Engineering specialty), represents a break with the CS discipline and an assumption of the socio-technical nature of the SE, and its inevitable convergence with the IS discipline (Kurbel, 2008).

In our approach PMS and DAI courses intend to provide students an environment of software development projects similar to an environment in organization context. They introduce engineering and management software techniques to collect and specify the requirements of the software development projects. They also deal with the software lifecycle management issues. These two courses are mainly responsible for software development methodologies and software project planning teaching.

The ACSI and GPSI courses intend to instill in students an engineering approach to information systems development. This attitude derives from the SE/IS convergence mentioned above and is materialized in the business solutions development study (enterprise business applications). Thus, the engineering and requirements management techniques and modeling and systems architecture are complemented and the maturity and software processes improvement issues are systematized, as well as the processes management and project development.

The Department of Information Systems (DSI) of the School of Engineering (EEUM) of the University of Minho, where this research is performed, offers an educational portfolio in the Information Systems and Technologies (IST) area that covers all levels of higher education. Initial training of IST professionals is achieved through the integrated Master in Engineering and Management of Information Systems (MIEGSI). This is the main master course that has PMS, DAI, ACSI and GPSI courses in its curriculum. The challenges of keeping up with the constant evolution of IST professional training and the demands of adjusting higher education programs to the new principles and rules for higher education have been addressed through several changes in the program structure of this Master's degree. The MIEGSI results from the Information Systems and Technologies BSc degree and the Engineering and Management of Information Systems MSc degree combination. These courses were the result of the adaptation (appropriateness) of the Information Management course to the higher education model established in 2006 (the Bologna Process).

Although the current *Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems* is IS 2010 (ACM-Curricula-IS, 2010), MIEGSI was originally structured according to the previous version of the standard, the IS 2002 (ACM-Curricula-IS, 2002).

Based on knowledge of the areas and their comparison we are able to infer that the MIEGSI is a course in the IS area, but with less depth in topics of *Information Technology* curriculum area than suggested by IS 2002. The PMS and DAI curricular units deal with some topics in great depth whose inclusion in ISBOK (*Information Systems Body of Knowledge*) has its origin in the SWEBOK (*Software Engineering Body of Knowledge*) (Abran, Bourque, Dupuis, Moore, & Tripp, 2004). These courses belong to a curriculum plan of IS and not CS or SE. The main difference is on the software organizational nature focus (business or enterprise software applications), in conjunction with the strategy and culture of the enterprise, the existence of a great diversity of requirements sources (from traditional business stakeholders to contexts in which the architecture of information systems already appears rudimentarily conceived) and the coexistence with professionals with very different perspectives on ITs (functional consultants, information systems architect, manager of technology infrastructure, ...), all of which require a large capacity of flexibility in the implementation and management of development processes.

The publication of IS 2010 for updating the IS 2002 model resulted from the natural changes of the technological and industrial practices that the domain has been permanently subject to from its beginnings. The IS 2010 model introduced some adjustments in the courses recommended as mandatory (core courses) and significantly increased the range of elective courses. The PMS course fulfill the stipulated of *IS 2010.4 IS Project Management* course. In the case of DAI course, its syllabus covers what is stipulated in *IS 2010.6 Systems Analysis and Design* and *Application Development* courses. This last course, despite being classified as elective type, appears to be referenced as mandatory for all profiles of professional training (career track) considered by IS 2010. Notwithstanding the structural and programmatic adjustments, the author of this chapter considers that the syllabus of MIEGSI is aligned with the recommendations of IS 2010.

In the case of ACSI and the GPSI courses from MIEGSI integrated master, the study of alignment with the curriculum frameworks requires analyzing the MSIS 2006 (ACM-Curricula-MSIS, 2006). The framework belongs to the *Computing Curricula* program, which in the AIS (Association for Information Systems) leadership focuses on post-graduate education in IS. This framework presents a strict balance between curricular areas of *IT Technology* and *IS Management* treated in mandatory courses which recommends.

The ACSI course is aligned with the *MSIS 2006.2 Analysis Modeling and Design* course. Following the recommendations of MSIS 2006, the ACSI course complement the training received, particularly in DAI course, on modeling and development cycles issues of the technology-based solutions. The GPSI course is aligned with the *MSIS 2006.5 Project and Change Management* course. Following the recommendations of MSIS 2006, the GPSI course complement the training received, particularly in PMS course, on management of the development process issues, introducing the whole socio-technical dimension of the IS and considering the projects as drivers of technological and organizational change.

The curriculum frameworks under the *Computing Curricula* program were developed with the explicit aim of being adopted in university education in the USA (United States of America) and Canada. However, the use of its recommendations outside this geographical context has proven to be a useful practice, with many more advantages than disadvantages. Given that the IT area (in its various forms) is still in its infancy (and therefore remains subject to rapid evolution, which makes it extremely difficult to keep up with the various curricular offerings) and that the abovementioned two countries are at the forefront of technological development, it is a recommended practice to regularly peruse what is being said about the training of their professionals.

Despite the enormous worldwide spread of the frameworks produced under the *Computing Curricula* program, there are countries that choose to follow different paths to develop their own frameworks as a result of a concerted effort within their universities. One example is Australia, where, in some universities, an approach has been adapted to training in IS with a specific curriculum (Tatnall & Burgess, 2009). The Australian approach has evolved from a perspective called "Information Management" and for decades was designated "Business Computing" (Retzer, Fisher, & Lamp, 2003).

Another example is Germany, which has developed a body of knowledge suitable for framing how the courses in IS should be taught (K. Kurbel, Krybus, & Nowakowski, 2013). In the German language, this area has historically been designated "Wirtschaftsinformatik" which over the years has been translated into English as "Business Informatics" or "Business Computer Science." Recently the accepted term for the IS area has been "Business & Information Systems Engineering."

The variability of possible approaches in teaching, not only in the IS sub-area, but, generally, in the IT (Computing) area suggests, sometimes, the use of accreditation curriculum and professional certification as a way to introduce some order in the training of professionals and to recognize, unequivocally, the specific skills that a professional must have in a set of sub-domains required by the industry.

OUR PROJECT BASED LEARNING APPROACH

This section describes how the integration of the four courses mentioned in the previous section has been managed, as well the solutions tried to promote educational cooperation in the teaching practices context based in projects. This integration has been carried out over the last four years. More precisely, it began in the academic year 2009/2010. This integration has been managed to allow minor changes over the years and to improve some relevant aspects. It is in this controlled context that we developed our PBL approach.

The *Bologna Declaration* (European Commission, 2010) was a political commitment to achieve in the short term, a clear set of objectives recognized as fundamental to the construction of the "European Higher Education Area" and to promote "European System of Higher Education" throughout the planet.

Integrated in the changes recommended by the *Bologna Process*, the ECTS is part of a set of procedures that support the new paradigm of organizing student-centered learning (and training objectives) and move from a traditional system curriculum based on the "juxtaposition" of knowledge to a system focused on developing broad curricular areas, defined in terms of training objectives to pursue. This argument reinforces the relevance of integrating a group of courses around global goals so that we can instill in students several skills in a way not comparable with skills obtained with the same courses in isolation mode. In ECTS, the work done by students in the subject area is expressed by a numerical value that takes into account the hours of student work, in their overall activities, including contact hours and hours spent on internships, projects, works in ground study and evaluation. The entire work done by the student in each course is expressed in ECTS credits and each ECTS credit corresponds to a total of 28 hours per semester.

The ECTS system is suited to changes in training, mainly in the development and adoption of: (1) new learning methodologies (more active and participatory), (2) horizontal capabilities and skills (learning to think, learning to learn, learning to teach), (3) specific skills of the profession, (4) general skills (communication capabilities, integration team, leadership, innovation and adaptation to change). All these dimensions were considered and are thought to be adequately incorporated in the joint project that integrates the four courses of our approach.

In many classrooms, learning is a passive activity. Students take notes during lectures and repeat the same information in the exams. When students read a chapter indicated by the teacher and they answer questions about that, the answers can be found in the chapter and are already known. Even in more experimental areas, the teachers rarely allow to students discover principles themselves; instead, the teachers present laws and techniques, and then they build exercises where students simply practice what they have been taught. This teaching is essentially based on the transmission of knowledge. The *Bologna Process* suggests switching to a school based on the students' work and the effective acquisition of skills, however, it should not put into question a proper proportion of more traditional activities also dedicated to the "simple" transmission of knowledge.

Although the *Bologna Process*, apparently "censors" the "teaching based on the transmission of knowledge," what is called contact time is no more than a series of moments in which teachers and students synchronize spatially to exchange knowledge with the perspective of developing, in students, certain skills. The *Bologna Process* should not avoid the transmission of knowledge between teachers and students, but rather aim to go further, catalyzing the emergence of new skills in students during the transfer of knowledge process. In this context, the transmission of knowledge is desirable and beneficial, so the pedagogical action in circumstances where every student has limited opportunities to interact with teachers does not foresee success. It is in the laboratory classes that transcendence of "skills transfer" can occur if each student is given the opportunity to receive knowledge. Consequently, the number of students spatio-temporally synchronized with the teachers in laboratory activities must be carefully determined, taking into account the expected level of depth of the "quasi-tangible" learning outcomes.

It was based on this change promoted by the *Bologna Process* that in the academic year 2006/07, it was decided to adopt teaching and learning practices based on the PBL (Barrows & Tamblyn, 1980) principles in the DAI course (since it is DAI course which catalyzes the main project work that integrates other courses). The teaching and learning context that PBL facilitates has been shown to be appropriate to the group of courses that integrate our environment (approach), where we want students to develop real skills in the production of technological artifacts imbued with a spirit of great rigor and methodological and procedural awareness and not simply to reproduce texts and definitions held in any book or manual timelessly accepted on any shelf or in any drawer. What drives the student to want to persist in school learning has to do mainly with the way we create and organize educational environments and all activities that we develop there.

Solving a problem according to the PBL approach requires the participation of students. The teacher helps and advises, but does not drive. Learning becomes an act of discovery as students examine the problem by investigating its base, analyze possible solutions, develop proposals and produce a final result. This active learning is not only more interesting and committed to the students but also develops a greater understanding of the syllabus since the students themselves seek information and then use it in an active way with the skills they already hold to complete the project. This way of organizing the teaching and learning activities for skills development promoted by the four courses is considered appropriate. Svinicki & McKeachie (2011) argue that "problem-based education is based on the assumptions that human being evolved as individuals who are motivated to solve problems, and that problem solvers will seek and learn whatever knowledge is needed for successful problem solving. Thus if an appropriate realistic problem is present before study, students will identify needed information and be motivated to learn it. However, as in introducing any other method, you need to explain to students your purposes" (Svinicki & McKeachie, 2011).

PBL requires the realization by students that learning lies in the prosecution of skills, not with the teacher to tell them that they are right, but based on experimentation with the artifacts and documents that they produce. In cooperative learning promoted by PBL, students learn from each other and they work together to develop the project. This aspect is extremely important in the context of our group of courses, in which there is the involvement of students with different academic degrees and maturity. In PBL, students grow more thoughtful and are harder working than in exercises that require rote memorization. In our approach, the emulation of a real project (with a real client) forces the students to learn from a variety of different sources and to make decisions based on their own research. This process allows students to achieve more advanced levels of cognitive skills, research skills and problem solving skills.

The PBL's real academic goal is not to develop a final response to the project. The students do not find just one true answer to the problem that, instantly, they agree can be the "correct" solution. Instead, the real learning occurs through the process of solving a problem: thinking through various steps, investigating the subjects and developing one solution. With the continuing explosion of knowledge and the pace of technological change, the universities cannot continue to provide all the information to the students that they need for their lives. Increasingly, the most important skill that the university can teach students is to learn by themselves. Within the group of courses integrated in our approach, this issue arises repeatedly when we want students to understand by themselves that throughout their careers will have to learn how to use and design new development processes, notation models, standards, paradigms, frameworks, management, process improvement and project standards.

The recognition of the effectiveness of the PBL approach in engineering teaching has resulted in several initiatives (The Higher Education Academy, 2003; University of Nottingham, 2003), including scientific projects of a pedagogical nature, in order to produce guidelines for the teaching and learning activities organization under PBL principles. In some forums, the use of PBL in engineering teaching has adopted the *Project-Led Engineering Education* (PLEE) designation (Powell, Powell, & Weenk, 2003).

The project that is presented every academic year to the PMS and DAI students requires energetic learners. Nobody will give them all the necessary information, nor will the answers be found in the books. To solve this problem requires that students "discover complaints," "investigate the reasons" and develop the best way to resolve the situation. Thus, our approach, which is based on this project, presents some crucial features for creating a context of enormous catalyzing of the phenomena of teaching and learning, namely:

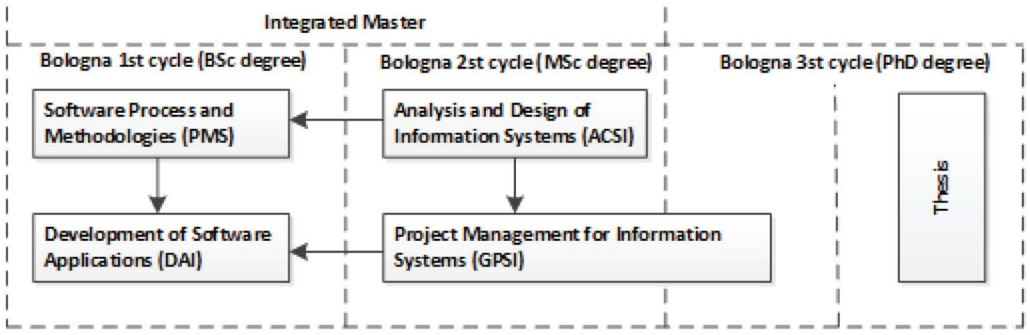
1. **Real client:** The existence of a real client, who interacts with, gathers and receives students in his organization, promotes in students the ability to feel, in practice, the difficulty of organizational software development with incomplete information and systematic doubts from the client, but with explicit business support needs (Trendowicz, Heidrich, & Shintani, 2011). It also allows students to understand how the customer thinks and evaluates the effort put into the development of the solution (Burge & Troy, 2006).
2. **Project proposal versus project:** In the first semester, in the PMS course, the project proposal phase is created, in which feasibility studies are designed and carried out, with some initial incursion into requirement elicitation. This phase ends with a project proposal elaboration, including time and cost estimates and an initial definition of generic features of the solution. This separation between the project proposal phase (emulated in PMS course) and the project phase (emulated in DAI course) allows the students to understand the different requirements and planning approaches that they are required to adopt in each of those circumstances and perceive the difference between a more commercial nature (required in the project proposal phase) and a more technically oriented approach (essential in the implementation phase of the project) (Brazier, Garcia, & Vaca, 2007).
3. **Large Teams:** The high dimension teams allow the recreation of the complexity of an industrial context in terms of the multiplicity of tasks to be performed (Blake, 2005; Chaczko, Davis, & Mahadevan, 2004) and the enormous need for interaction between the different roles, promoting the development of skills of an inter-personal nature (Slaten, Droujkova, Berenson, Williams, & Layman, 2005). The development of the *soft skills* is one of the great gains acquired by students as a result of their involvement in the project.
4. **Rigor in software process development:** The adoption of a configurable software process development permits the instillation of awareness and rigor in how students decide

the management and implementation of the project (Suri & Sebern, 2004). Actually, this procedural rigor results from the RUP framework adoption with the reduced model of roles (Borges, Monteiro, & Machado, 2011), the obligation to follow the CMMI model practices and to adopt the PMBOK (*Project Management Body of Knowledge*) (PMI, 2008) and SWEBOK bodies of knowledge. With these standards, the relationship between the project management and the software process development adopted becomes explicit, allowing sensitization of the students to the dire need to own specific methodological skills in the area in order to be able to manage projects in the field of technology and information systems. Additionally, it allows the creation of a true perception of the relationship between the structures of the solution implementation and the requirements that gave rise to it (Burgstaller & Egyed, 2010), as well as between product quality and the rigor of process practices adopted by the team.

5. **Complexity of the solution:** The project can create conditions in the educational context where students are engaged in designing and building medium-size software solutions based on requirements from the "real world." This makes for the experience of conceiving an architecture which is methodologically plausible (Naveda, 1999), while developing the students' sensitivity to the effectively effort needed to conceptualize artifacts with similar complexity to those developed in industrial contexts (Wohlin, 1997).
6. **Others:** In every academic year a mechanism or new feature in project work is introduced in the second semester that cumulatively add some novelty to the operational mode of the previous year. Over the years, the following features or mechanisms were introduced (not in chronological order): alignment with SWEBOK, alignment with PMBOK, RUP model, internal evaluation and promotion of human resources, CMMI level 2 practices, CMMI level 3 practices, real client, final presentation with a commercial focus, formalization of the product delivery, outsourcing, technological interoperability between partial solutions, consulting of outside services, use of patterns, formal documentation with RUP templates. It is under study by the teaching team, how the following mechanisms or features will be adopted over the next academic years: agile practices, reinforcement and hiring human resources, competition between enterprises and corporate bankruptcy.

Over the years it has only been possible to cumulatively manage all these features and mechanisms because the teaching staff is stable, cohesive and aligned in the way it organizes and engages with students who annually attend the four courses (about a hundred students in PMS and DAI and from about three-and four dozen students in ACSI and GPSI). The curriculum integration and the pedagogical cooperation among the four courses have also been decisive in making possible the management of all these educational processes that converge in the laboratory classes in the DAI course, since it consists of the place of convergence by the students from both levels of education that functionally associate in the project (see Figure 1): (1) the ACSI students provide external services (regarding to the functional consultant or senior analyst role) to the PMS students and GPSI students provide external services (regarding to the project facilitator, senior project manager or process engineer role) to the DAI students; (2) the PMS students emulate phenomena studied by ACSI students and DAI students emulate phenomena studied by GPSI students. Often ACSI and GPSI students choose for their dissertations some of the themes that they dealt when they worked with PMS and DAI students.

Figure 1. Integration among courses


Integrated Master
Bologna 1st cycle (BSc degree)
Bologna 2st cycle (MSc degree)
Bologna 3st cycle (PhD degree)
Software Process and Methodologies (PMS)
Analysis and Design of Information Systems (ACSI)
Development of Software Applications (DAI)
Project Management for Information Systems (GPSI)
Thesis

Recently, some of the PhD students in the SEMAG (*Software Engineering and Management Group*) research group (SEMAG, 2001) also began to engage with the students of the four courses, in order to carry out scientific studies supported in educational context experiments and this work has given rise to international publications (some involving ACSI and GPSI students) (Monteiro, Borges, Machado, & Ribeiro, 2012; Monteiro et al., 2013). Thus, this group of four courses is simultaneously a space of pedagogical and scientific innovation from which all stakeholders benefit (Siqueira, Barbaran, & Becerra, 2008). It is our perception that almost all stakeholders in this huge process (students, teaching staff and researchers) feel "pleasure" with the involvement in the project (Robert L. Glass, 2007; R. L. Glass, 2007), despite the great dedication and tremendous effort that is demanded from all involved.

RESULTS OBTAINED FROM OUR PBL APPROACH

The real-world project/study approach to teaching software engineering has been successful thus far. It has helped to motivate the teams and to encourage development of higher quality products by the teams. The teams took seriously the importance of the problems that they were helping to solve. The approach teaches inexperienced graduate students many principles of software engineering and software verification and validation that they could apply to newly gained jobs and/or subsequent courses (Hayes, 2002).

Our PBL approach began to be effectively implemented in the academic year 2009/2010, although some well-controlled trials had been tested in previous years. The real client collaboration in the software project just started in that academic year. This client was located in the region of our university. The real client activity area was quite diverse. In our previous software projects we had collaborations with a factory enterprise, a non-profit institution of social solidarity, a professional handball team and a professional football team.

Over the past four years we have had, approximately, one thousand students involved in the four educational syllabuses. In each academic year, we had approximately 150 students in PMS and DAI courses and 100 students in ACSI and GPSI courses.

Evaluation of Student Teams in Our PBL Approach

It is not easy to teach the syllabus of the four courses because of their considerable size and a lack of conditions revealed by the students for realization of topics with a more abstract nature. In fact, this group of courses has a considerably high level of requirement, so the methodologies or teaching strategies to be adopted should be in accordance with the level of depth of the topics.

The students should be able to use the study objects to perform the tasks. In the case of the DAI course, this level of learning depth requires a deep involvement by the students in practical classes (in the form of exercises proposed and solved in class), complemented by the development of a software project. This work begins in the first semester under the PMS course, in groups of about five students. These groups of students emulate the project proposal context in which they perform an initial analysis of requirements and time and cost planning of the project proposal solution for a real client with whom the teaching staff establishes an annual academic collaboration partnership (Ali, 2006; Kornecki, Khajenoori, Gluch, & Kameli, 2003). In the DAI course, in the second semester, the work project is performed by groups of about 15 students. These groups of students emulate the project implementation context of a proposed solution to the same real client, which they perform a comprehensive analysis and design of the problem, construction, testing and delivery of a software solution coded in an object-oriented language (*Java* or *C#*) with relational databases (*SQL Server* or *MySQL*) and with interfaces for the Web. In both courses, student groups follow the RUP model (Bergandy, 2008) and they use the UML notation extensively. The teams use the laboratory classes to meet in the presence of teachers. In these weekly meetings, the teachers monitor the progress of the project work.

In the case of ACSI and GPSI courses, the learning outcomes with higher depth level are achieved through the involvement of students with groups of PMS and DAI courses, respectively, promoting play roles with responsibility and a high level of maturity, such as the functional consultant, senior analyst, project facilitator, senior project manager or process engineer. These two courses have different learning outcomes with deep levels corresponding to some of the emerging topics addressed and discussed in lectures, which prepare the students with a good understanding of a set of problems that currently emerge in the area.

The learning outcomes with a more advanced level are achieved through the use of a scientific literature searches in order to study in detail some complex themes that arise in the engagement context by the ACSI and GPSI students when they participate in the project work performed by PMS and DAI students. The two semiannual project works instill a hands-on training dynamic that is essential for the two pairs of courses operation as a whole (Broman, 2010; Port & Boehm, 2001): PMS and ACSI courses in the first semester and DAI and GPSI courses in the second semester.

The university regulation concerning the evaluation of courses was changed in 2004. From that year it became possible to evaluate a student that "only" executes a project and thus dispense with the realization of an exam. Thus, the evaluation system adopted allowed the organization of all activities of the four courses around an annual educational project. This project is the integrator of all the topics listed in the syllabus and it follows a student time management approach that is able to facilitate the effective development of skills necessary for evidencing learning outcomes stipulated (Stiller & LeBlanc, 2002). Since the academic year 2006/07 this group of courses has adopted this evaluation system, which provides a much more effective management of students activities inside and outside of the university physical spaces and in a horizon that it extends over the two semesters of the academic year, according to the PBL approach.

Since DAI is the core course in our approach then we will describe in some detail the evaluation system of this curricular unit. This evaluation scheme encourages the incremental demonstration of the partial learning outcomes over the semester.

Between the two semesters in which annual project is developed, the second semester is the one that concentrates the largest effort and diversity of activities and in this sense DAI is a curricular unit of 10 ECTS. The student evaluation in DAI is carried out based on the activities undertaken by various teams within the semiannual project work in five *assessment milestones*. Each one of the five *assessment milestones* is associated with partial evaluation, four are team evaluations, and one is an individual evaluation. However, it is systematic that several students in a team obtain different classifications in the four team *assessment milestones*, since the individual score calculation results from a plurality of weighting factors from three sources of information (Clark, 2005), such as the teachers traditional evaluation, client evaluation and peer evaluation (other students of the team).

Work Projects Developed in ACSI and GPSI Courses

By academic year, the reports developed in ACSI and GPSI are edited in two separated documents which are assigned with ISSN (International Standard Serial Number). These documents contain all the student project works of both courses. These project works can be a report or a scientific paper that describes all the work performed by the students. These documents serve as consultation reference for future students. These students can find the difficulties, limitations, the positive and negative aspects and lessons learned and experienced by colleagues of the previous years. Table 1 was created based on a detailed analysis of these documents. The table presents a mapping between the SWEBOK Knowledge Areas (KA) and the work projects developed by the ACSI and GPSI students.

As we can see in Table 1, in the 2010/2011 and 2011/2012 academic years, the 140 students enrolled in the ACSI and GPSI courses developed 130 work projects in different SWEBOK KAs. Although the majority of work projects are developed individually, some of them involve teams of 2 or 3 students.

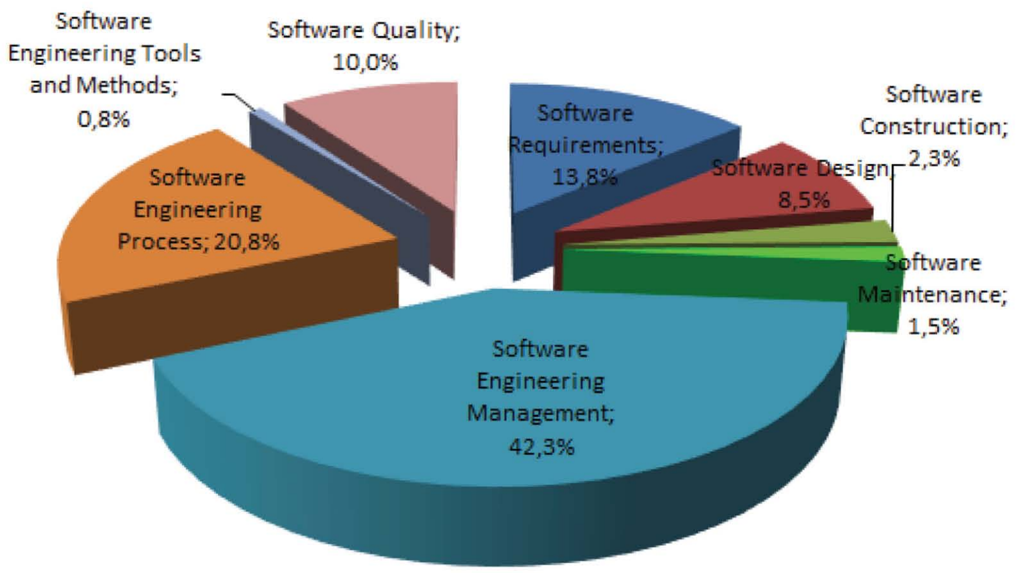
Figure 2 shows that most of the project works are developed in the *Software Engineering Management* area, representing 42.3% of the total. The high percentage of the project works in this KA is justified by the involvement of ACSI and GPSI MSc students as a consultant/facilitator role of the DAI development teams (BSc students).

Another conclusion obtained by the Table 1 analysis is the total absence of project works in the *Software Testing* and *Software and Configuration Management* areas. Normally the tests are

Table 1. ACSI and GPSI work projects by SWEBOK knowledge area

SWEBOK Knowledge Area	**Academic Year**				**Sum**
	2010/2011		**2011/2012**		
	ACSI	GPSI	ACSI	GPSI	
Software Requirements	2	2	8	6	18
Software Design	3	0	7	1	11
Software Construction	0	0	1	2	3
Software Testing	0	0	0	0	0
Software Maintenance	0	0	1	1	2
Software Configuration Management	0	0	0	0	0
Software Engineering Management	11	7	11	26	55
Software Engineering Process	6	0	13	8	27
Software Engineering Tools and Methods	0	1	0	0	1
Software Quality	2	0	3	8	13
Sum	24	10	44	52	130

Figure 2. ACSI and GPSI work projects by SWEBOK KA



performed by the DAI students, those who develop the software applications, and until now, the MSc students have not expressed interest in this KA. Regarding *Software and Configuration Management,* the absence is justified by the fact that we are working in an educational context and some activities of that KA are not performed, such as, software configuration identification, software configuration control, software configuration status accounting, software configuration auditing, and software release management and delivery. It is also worth noting the low number of project works in *Software Construction* and *Software Engineering Tools and Methods* KAs.

Table 2 and Table 3 show the topics and subtopics of the SWEBOK KA covered by ACSI and GPSI Work Projects respectively. The numbers preceding the names of the topics and subtopics are those in SWEBOK document.

From Table 2 and Table 3, we can infer that seven and eight of the ten SWEBOK KAs are covered by the ACSI and GPSI work projects respectively. We just can find one GPSI work project in the *Software Engineering Tools and Methods* area. Specifically, this project work involved the *Microsoft Project Server 2007* key features study, through two visions: one as a project manager and other as system administration. Despite the topics and subtopics diversity covered by the work projects, we cannot infer that there is a great variability of subjects between the two courses.

FUTURE RESEARCH DIRECTIONS

It is a well-accepted fact that a competent software engineer requires a wide variety of skills such as managerial, engineering, team working and communication. The problem is how to teach these skills to students within the classroom. One approach to solving, or at least to mitigating this problem, is PBL. There are many reasons for incorporating real industry projects: increased student motivation and confidence; the ability for students to explore in-depth IT areas not covered in the curriculum or only covered superficially; the development in students of increased problem-solving and critical thinking skills, communication skills and business insight.

Table 2. Topic and subtopic of the SWEBOK KA covered by ACSI work projects

SWEBOK KA	ACSI Course	
	SWEBOK Topic	**SWEBOK** Subtopic
Software Requirements	1. Software Requirements Fundamentals	1.1. Definition of a Software Requirement
		1.2. Product and Process Requirements
		1.3. Functional and Non-functional Requirements
	2. Requirements Process	2.3. Process Support and Management
	4. Requirements Analysis	4.1. Requirements Classification
		4.2. Conceptual Modeling
	5. Requirements Specification	5.3. Software Requirements Specification
	7. Practical Considerations	7.2. Change Management
		7.4. Requirements Tracing
Software Design	1. Software Design Fundamentals	1.3. Software Design Process
	3. Software Structure and Architecture	3.1. Architectural Structures and Viewpoints
		3.2. Design Patterns (microarchitectural patterns)
Software Construction	3. Practical considerations	3.2. Construction Languages
Software Maintenance	4. Techniques for Maintenance	4.3. Reverse engineering
Software Engineering Management	2. Software Project Planning	2.1. Process planning
		2.4. Resource allocation
		2.5. Risk management
	3. Software Project Enactment	3.2. Supplier contract management
		3.4. Monitor process
		3.5. Control process
Software Engineering Process	1. Process Implementation and Change	1.1. Process infrastructure
		1.2. Software process management cycle
	2. Process Definition	2.1. Software life cycle models
		2.2. Software life cycle processes
		2.3. Notations for Process Definitions
	4. Process and Product Measurement	4.1. Process measurement
		4.2. Software product measurement
Software Quality	1. Software Quality Fundamentals	1.3.Models and quality characteristics

As a part of our approach, the teaching team is currently studying how the following mechanisms or features will be adopted over the next academic years: agile practices, reinforcement and hiring of human resources, competition between enterprises and corporate bankruptcy. The inclusion of these features and mechanisms should be done with a great deal of reflection and sensitivity so as not to collide with ethics, morality, privacy and pedagogical issues. Another aspect to consider in this type of environmental change is the internal regulations of the university.

Another important challenge in the near future is to develop an approach that allows, in the educational context, possibly with industrial cooperation, the planning, design and implementation of software projects in geographically distributed contexts and locations in order to strive for globalization.

Table 3. Topic and subtopic of the SWEBOK KA covered by GPSI work projects

SWEBOK KA	GPSI Course	
	SWEEBOK Topic	**SWEEBOK** Subtopic
Software Requirements	1. Software Requirements Fundamentals	1.3. Functional and Non-functional Requirements
	2. Requirements Process	2.1. Process Models
		2.3. Process Support and Management
	3. Requirements Elicitation	3.1. Requirements Sources
		3.2. Elicitation Techniques
	4. Requirements Analysis	4.4. Requirements Negotiation
	5. Requirements Specification	5.3. Software Requirements Specification
Software Design	5. Software Design Notations	5.1. Structural Descriptions (static view)
		5.2. Behavioral Descriptions (dynamic view)
Software Construction	3. Practical considerations	3.1. Construction Design
		3.2. Construction Languages
		3.3. Coding
		3.4 Construction Testing
Software Maintenance	4. Techniques for Maintenance	4.3. Reverse engineering
Software Engineering Management	2. Software Project Planning	2.4. Resource allocation
		2.5. Risk management
		2.6. Quality management
		2.7. Plan management
	3. Software Project Enactment	3.1. Implementation of plans
		3.2. Supplier contract management
		3.4. Monitor process
		3.5. Control process
		3.6. Reporting (RUP Roles)
Software Engineering Process	1. Process Implementation and Change	1.1. Process infrastructure
	2. Process Definition	2.2. Software life cycle processes
	4. Process and Product Measurement	4.1. Process measurement
		4.2. Software product measurement
Software Engineering Tools and Methods	1. Software Engineering Tools	1.7. Software Engineering Management Tools
Software Quality	1. Software Quality Fundamentals	1.3.Models and quality characteristics
	2. Software Quality Management Processes	2.1.Software Quality Assurance
		2.2.Verification & Validation
	3. Practical Considerations	3.4.Software Quality Measurement

Finally it is important to emulate a software process development environment that allows for using a plurality of reference standard quality models. This work can drive us to implement an approach where the teams, in an educational context, must follow this new environment.

CONCLUSION

The approach described in this chapter to teach topics of the engineering and management process of the software process development in university courses in the IS area aims to train professionals for the industry. These students, professionals in the near future, must be able to apply the most modern methodological approaches in the analysis and design of IS based on software, as well as manage the corresponding projects and development processes, in close cooperation with the most demanding quality and maturity procedural reference models. This dual training of students in engineering approaches and methodologies and in techniques and management methods are complemented with interpersonal skills. The dynamics promoted by the teaching-learning project promoted by the integration of the four annual courses develops management, engineering and personal skills. This tripartite training promoted by our approach instills in students an attitude and not only technological learning, in the SE context. The vision of *Engineering and Management Software* applied to IS problems is what industry seeks to build. So that engineering teams are increasingly able to intervene in organizations.

Using case studies, even with awareness of the limitations of possible inferences and simplifications when compared with empirical software engineering approaches, the students will continue their studies through various courses, capitalizing on experience, awareness, insight and ability to avoid paths of less desirable decisions. By the end of their studies, trainees will be ready to integrate easily in the industrial world with developed maturity and valuable experience.

An engineer cannot be just a "mere" specialist: it is necessary to combine technical skills with the human and social dimensions. Our students should be prepared to live and work as global citizens, understand how engineers contribute to society. They should be aware that it is not enough to be technically excellent, because there are other dimensions to consider. Our approach allows delivering some important skills: (1) a basic understanding of business processes; (2) a product development with high-quality concerns; (3) know-how to conceive, design, implement and operate medium-size complexity systems and (4) communicative, initiative/leadership, teamwork, analytical and problem solving and personal abilities.

REFERENCES

Abran, A., Bourque, P., Dupuis, R., Moore, J. W., & Tripp, L. (2004). *Guide to the software engineering body of knowledge (SWEBOK)*. Los Alamitos, CA: IEEE Computer Society Press.

ACM-Curricula-IS. (2002). *IS 2002: Model curriculum and guidelines for undergraduate degree programs in information systems. Association for Computing Machinery (ACM), Association for Information Systems (AIS), Association of Information Technology Professionals*. AITP.

ACM-Curricula-IS. (2010). *IS 2010: Curriculum guidelines for undergraduate degree programs in information systems. Association for Computing Machinery (ACM), Association for Information Systems*. AIS.

ACM-Curricula-MSIS. (2006). *Model curriculum and guidelines graduate degree programs in information systems. Association for Information Systems*. AIS.

Ali, M. R. (2006). Imparting effective software engineering education. *ACM SIGSOFT Software Engineering Notes*, *31*(4), 1–3. doi:10.1145/1142958.1142960

Avison, D., & Wilson, D. (2001). A viewpoint on software engineering and information systems: What we can learn from the construction industry? *Information and Software Technology*, *43*(13), 795–799. doi:10.1016/S0950-5849(01)00186-0

Barrows, H. S., & Tamblyn, R. H. (1980). *Problem-based learning: An approach to medical education*. New York: Springer.

Bergandy, J. (2008). Software engineering capstone project with rational unified process (RUP). In *Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference*. New York: ASEE/IEEE.

Birk, A., Heller, G., John, I., Schmid, K., von der Massen, T., & Muller, K. (2003). Product line engineering, the state of the practice. *IEEE Software*, *20*(6), 52–60. doi:10.1109/MS.2003.1241367

Blake, M. B. (2005). Integrating large-scale group projects and software engineering approaches for early computer science courses. *IEEE Transactions on Education*, *48*(1), 63–72. doi:10.1109/TE.2004.832875

Borges, P., Monteiro, P., & Machado, R. J. (2011). Tailoring RUP to small software development teams. In *Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications* (pp. 306–309). IEEE.

Brazier, P., Garcia, A., & Vaca, A. (2007). A software engineering senior design project inherited from a partially implemented software engineering class project. In *Proceedings of the 37th ASEE/IEEE Frontiers in Education Conference* (pp. F4D–7). IEEE.

Broman, D. (2010). Should software engineering projects be the backbone or the tail of computing curricula? In *Proceedings of the 23rd IEEE Conference on Software Engineering Education and Training*. Pittsburgh, PA: IEEE.

Burge, J., & Troy, D. (2006). Rising to the challenge: Using business-oriented case studies in software engineering education. In *Proceedings of the 19th Conference on Software Engineering Education and Training* (pp. 43–50). IEEE.

Burgstaller, B., & Egyed, A. (2010). Understanding where requirements are implemented. In *Proceedings of the 26th IEEE International Conference on Software Maintenance* (pp. 1–5). IEEE.

Chaczko, Z., Davis, D., & Mahadevan, V. (2004). New perspectives on teaching and learning software systems development in large groups. In *Proceedings of the 5th International Conference on Information Technology Based Higher Education and Training* (pp. 409–414). IEEE.

Clark, N. (2005). Evaluating student teams developing unique industry projects. In *Proceedings of the 7th Australasian Computing Education Conference* (pp. 21–30). Australian Computer Society.

Dutoit, A. H., McCall, R., Mistrik, I., & Paech, B. (2006). Rationale management in software engineering: Concepts and techniques. In A. H. Dutoit, R. McCall, I. Mistrik, & B. Paech (Eds.), *Rationale management in software engineering*. Springer. doi:10.1007/978-3-540-30998-7_1

Engle, C. B. (1989). Software engineering is not computer science. In N. Gibbs (Ed.), *Software engineering education* (Vol. 376, pp. 257–262). New York: Springer. doi:10.1007/BFb0042363

European Commission. (2010). *The Bologna process - Towards the european higher education area*. Retrieved 08-07-2013, 2013, from http://ec.europa.eu/education/higher-education/bologna_en.htm

Finkelstein, A. (2011). Ten open challenges at the boundaries of software engineering and information systems. In H. Mouratidis, & C. Rolland (Eds.), *Advanced information systems engineering* (Vol. 6741, pp. 1–1). New York: Springer Berlin Heidelberg. doi:10.1007/978-3-642-21640-4_1

Fung, R. Y. K., Tam, W. T., Ip, A. W. H., & Lau, H. C. W. (2002). Software process improvement strategy for enterprise information systems development. *International Journal of Information Technology and Management*, *1*(2-3), 225–241. doi:10.1504/IJITM.2002.001198

Glass, R. L. (2007). Is software engineering fun? *IEEE Software*, *24*(1), 96–95. doi:10.1109/MS.2007.18

Glass, R. L. (2007). Is software engineering fun? Part 2. *IEEE Software*, *24*(2), 104–103. doi:10.1109/MS.2007.46

Gregg, D., Kulkarni, U., & Vinzé, A. (2001). Understanding the philosophical underpinnings of software engineering research in information systems. *Information Systems Frontiers*, *3*(2), 169–183. doi:10.1023/A:1011491322406

Hayes, J. H. (2002). Energizing software engineering education through real-world projects as experimental studies. In *Proceedings of the 15th Conference on Software Engineering Education and Training* (pp. 192–206). IEEE.

Hellens, L. A. (1997). Information systems quality versus software quality a discussion from a managerial, an organisational and an engineering viewpoint. *Information and Software Technology*, *39*(12), 801–808. doi:10.1016/S0950-5849(97)00038-4

Higher Education Academy. (2003). *PBLE (project based learning in engineering)*. Retrieved 09-07-2013, 2013, from http://www.heacademy.ac.uk/resources/detail/resource_database/SNAS/PBLE_Project_Based_Learning_in_Engineering

Jayaratna, N., & Sommerville, I. (1998). The role of information systems methodology in software engineering. *IEE Proceedings. Software*, *145*(4), 93–94. doi:10.1049/ip-sen:19982193

Karunasekera, S., & Bedse, K. (2007). Preparing software engineering graduates for an industry career. In *Proceedings of the 20th Conference on Software Engineering Education & Training* (pp. 97–106). IEEE.

Kornecki, A. J., Khajenoori, S., Gluch, D., & Kameli, N. (2003). On a partnership between software industry and academia. In *Proceedings of the 16th Conference on Software Engineering Education and Training* (pp. 60–69). IEEE.

Kurbel, K., Krybus, I., & Nowakowski, K. (2013). *Lehrstuhl für wirtschaftsinformatik*. Retrieved 2013-07-05, 2013, from http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/uebergreifendes/

Kurbel, K. E. (2008). *The making of information systems: Software engineering and management in a globalized world*. Berlin, Germany: Springer. doi:10.1007/978-3-540-79261-1

McBride, N. (2003). A viewpoint on software engineering and information systems: Integrating the disciplines. *Information and Software Technology*, *45*(5), 281–287. doi:10.1016/S0950-5849(02)00213-6

Monteiro, P., Borges, P., Machado, R. J., & Ribeiro, P. (2012). A reduced set of RUP roles to small software development teams. In *Proceedings of International Conference on Software and System Process* (pp. 190–199). IEEE.

Monteiro, P., Machado, R., Kazman, R., Lima, A., Simões, C., & Ribeiro, P. (2013). Mapping CMMI and RUP process frameworks for the context of elaborating software project proposals. In *Software quality: Increasing value in software and systems development* (Vol. 133, pp. 191–214). Berlin: Springer. doi:10.1007/978-3-642-35702-2_12

Naveda, J. F. (1999). Teaching architectural design in an undergraduate software engineering curriculum. In *Proceedings of the 29th ASEE/IEEE Frontiers in Education Conference* (Vol. 2, pp. 12B1–1). IEEE.

Nunan, T. (1999). *Graduate qualities, employment and mass higher education*. Paper presented at the HERDSA Annual International Conference. Melbourne, Australia.

Platt, J. R. (2011). Career focus: Software engineering. *IEEE-USA Today's Engineer*. Retrieved from http://www.todaysengineer.org/2011/Mar/career-focus.asp

PMI. (2008). *A guide to the project management body of knowledge* (4th ed.). Newtown Square, PA: Project Management Institute, Inc.

Port, D., & Boehm, B. (2001). Using a model framework in developing and delivering a family of software engineering project courses. In P*roceedings of the 14th Conference on Software Engineering Education and Training* (pp. 44–55). IEEE.

Powell, W., Powell, P. C., & Weenk, W. (2003). *Project-led engineering education*. Lemma Publishers.

Retzer, S., Fisher, J., & Lamp, J. (2003). Information systems and business informatics: An Australian German comparison. In *Proceedings of the 14th Australasian Conference on Information Systems* (pp. 1–9). Edith Cowan University.

SEMAG. (2001). *Software engineering and management group*. Retrieved 01-06-2013, 2013, from https://sites.google.com/a/dsi.uminho.pt/semag/

Shere, K. D. (1988). *Software engineering management*. Upper Saddle River, NJ: Prentice Hall.

Siqueira, F. L., Barbaran, G. M. C., & Becerra, J. L. R. (2008). A software factory for education in software engineering. In *Proceedings of the 21st Conference on Software Engineering Education & Training* (pp. 215–222). IEEE.

Slaten, K. M., Droujkova, M., Berenson, S. B., Williams, L., & Layman, L. (2005). Undergraduate student perceptions of pair programming and agile software methodologies: Verifying a model of social interaction. In *Proceedings of the Agile Development Conference* (pp. 323–330). IEEE.

Stiller, E., & LeBlanc, C. (2002). Effective software engineering pedagogy. *Journal of Computing Sciences in Colleges*, *17*(6), 124–134.

Suri, D., & Sebern, M. J. (2004). Incorporating software process in an undergraduate software engineering curriculum: Challenges and rewards. In *Proceedings of the 17th Conference onSoftware Engineering Education and Training* (pp. 18–23). IEEE.

Svinicki, M., & McKeachie, W. J. (2011). *McKeachie's teaching tips: Strategies, research, and theory for college and university teachers*. Wadsworth: Cengage Learning.

Tatnall, A., & Burgess, S. (2009). Evolution of information systems curriculum in an Australian university over the last twenty-five years. In A. Tatnall, & A. Jones (Eds.), *Education and technology for a better world* (Vol. 302, pp. 238–246). Berlin: Springer. doi:10.1007/978-3-642-03115-1_25

Tockey, S. (1997). A missing link in software engineering. *IEEE Software*, *14*(6), 31–36. doi:10.1109/52.636594

Trendowicz, A., Heidrich, J., & Shintani, K. (2011). Aligning software projects with business objectives. In *Proceedings of the 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement* (pp. 142–150). IEEE.

University of Nottingham. (2003). *PBLE (project based learning in engineering)*. Retrieved 09-07-2013, 2013, from http://www.pble.ac.uk/

Wohlin, C. (1997). Meeting the challenge of large-scale software development in an educational environment. In *Proceedings of the 10th Conference on Software Engineering Education and Training* (pp. 40–52). IEEE.

KEY TERMS AND DEFINITIONS

Assessment Milestone: It is a partial evaluation assigned by teachers' staff to the students. This event gives to the students an orientation about their effort in the software project development.

Engineering Skills: Know how to perform requirements engineering, analysis and design, implementation, quality control and software administration.

Managerial Skills: Know how to perform resource estimation and tracking, project scheduling and tacking, software development life cycle, risk management, configuration management, release management, quality assurance and traceability management.

Personal Skills: All communication, initiative/leadership, teamwork, analytical and problem solving and personal time abilities.

Project-Based Learning: Considered an alternative to paper-based, rote memorization, teacher-led classrooms. The benefits of these strategies implementation in the classroom including a greater depth of understanding of concepts, broader knowledge base, improved communication and interpersonal/social skills, enhanced leadership skills, increased creativity, and improved writing skills.

Rational Unified Process: It is an iterative software development process framework created by the Rational Software Corporation, a division of IBM. RUP is an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs.

SWEBOK: It is a *Software Engineering Body of Knowledge* guide created under the IEEE sponsorship in order to serve as a reference in issues considered relevant by the Software Engineering community.