Teaching Software Process Improvement and Assessment

Christiane Gresse von Wangenheim[1], Jean Carlo R. Hauck[1, 2]
[1]Federal University of Santa Catarina (UFSC)
Florianópolis/SC – Brazil
gresse@gmail.com
[2] DkIT – Dundalk Institute of Technology
Dundalk/Co. Louth, Ireland
jeanhauck@gmail.com

Abstract

In order to prepare students for careers as software process engineers, software engineering education needs to adopt innovative instructional designs to support effectively the learning of required knowledge and skills. In this paper, we propose a cross-course design for teaching software process improvement and assessment in a graduate course in combination with an undergraduate capstone project course adopting a constructivist approach. We applied the proposed course design and investigated its impact on learning effects, its adequacy and strengths and weaknesses by administering a pre-and post-test and applying a questionnaire at the end of the course. First evaluation results indicate a positive learning effect on students to develop competencies required for software process engineers as well as it successfully engaged both graduate and undergraduate students while providing a beneficial experience through their interactions.

Keywords

Software Process Improvement, CMMI, Software Process Assessment, Education, Software Engineering

1 Introduction

Software process improvement (SPI) is becoming more important each year in order to meet the challenge of complex software systems and an increasing demand for more reliable systems. By now a large number of software organizations have established software process improvement initiatives and many of them have been formally assessed [1]. Yet, given the broad range of approaches, methods and tools for SPI, organizations are struggling to find competent professionals, who are able to effectively engineer software processes around the organization [2]. In this context, an additional, distinct role is required, the Software Process Engineer (SPE) [3], who is responsible for the definition, assessment, establishment and maintenance of software processes, analysis of quality problems and support for the implementation of improvement suggestions [4]. These responsibilities are distinct enough from other software development or management tasks that responsibles need to have specific knowledge on SPI concepts and processes, software process capability/maturity models and standards, software process improvement and assessment methods as well as good interpersonal skills [2, 4]. And, although there have been significant advances during the last year in software engineering education, in general, e.g., through the development of the SWEBOK [5] and Software Engineering (SE) curriculum guidelines [6], less emphasis on SPI can be observed even on the graduate level [7, 8]. SPI content is more typically taught in professional training courses and/or formal preparation courses for professional certifications, such as, e.g., SEI's SEPM Certificate Programs (http://www.sei.cmu.edu/credentials/sepm.html#sepm) or the International Software Process Improvement Certification (ISPIC) (http://www.spinstitute.org/certification.htm). Therefore, it becomes important to provide opportunities for students to learn these required knowledge and skills also as part of formal education [7].

Another issue is the way in which SE courses are typically taught. Expository lessons are still the dominant instructional technique in, basically, all sectors of education and training [9]. While they are adequate to present abstract concepts and factual information, they are not the most suitable for higher-cognitive objectives aiming at the application and transfer of knowledge to real-life situations [10]. Thus, in order to improve SE education, a general trend is to emphasize “hands on” experience for the students related either to industry or a simulated environment [11. Yet, so far, there have been made only very few proposals for teaching SPI effectively recreating an authentic context in which software processes are engineered in the classroom.

At the Master Program in Applied Computer Science at the UNIVALI – Universidade do Vale do Itajaí in Brazil, a SPI course is being held for master students since 2006. In the beginning, the course concentrated on theoretical topics. A classical educational method had been adopted using expositive lectures, discussions and case study reviews. However, it seemed that the course did not successfully teach the content on the application level and did not motivate the students sufficiently. As a result students acquired a surface knowledge of basic topics, but had problems to apply them as well as to achieve higher cognitive levels as a basis for their research projects.

Thus, in 2009, the SPI course has been re-designed with the objective to increase the learning effect, specifically, on the application level and reinforcing the understanding of relevant concepts. A constructivist approach has been adopted, using situated learning and problem solving in an authentic context through group work and the nurturance of reflexivity and learning in an ill-structured domain. As, due to confidentiality reasons and unavailability of staff, no access to a real software organization was given in order to apply software process assessment (SPA) and improvement, the graduate SPI course has been combined with a capstone project in the undergraduate Computer Science course at the UNIVALI. This recreated a realistic context, which provided the opportunity for the graduation students to apply SPI concepts on the software projects being run in the undergraduate course. This cross-course design successfully engaged both graduate and undergraduate students, while providing a beneficial practical experience, which contributed to learning on the cognitive level of application.

2 Related Work

SE education has been received increased attention recently, and much progress has been made, principally by the development of the SWEBOK [5] as well as curriculum guidelines for SE courses [6]. And, although, the most common approach to teaching SE is the use of lectures, supplemented by laboratory sessions, tutorials, etc. [6], there have various articles published on educational strategies for SE education. Recent trends reflect a shift from objectivist learning, which views learning as the transmission of knowledge from the teacher to the learner, to constructivist learning, regarding learning less as the product of transmission than a process of active construction [6,13]. In this setting, diverse instructional designs and experiences on SE education have been published [11, 14], but, only very few focus specifically on SPI education. An exception is the experience reported by T. Dingsøyr et al. [15] on teaching SPI around an industrial case study based on lectures and group exercises. Another example is a graduate SE course to educate students on the basic concepts of SPE proposed by Hawker [3]. This course design is based on the OMG Software Process Engineering Metamodel and the IEEE Standard for Developing a Software Project Life Cycle Process as ways to model and compare process design alternatives and to provide mechanisms to assemble reusable process components into enactable processes. Other courses use the Personal Software Process (PSP) to teach software process improvement [16]. At the Ecole Polytechnique de Montreal a SE course held [17], carries out a project where students use a simplified version of the Trillium model to assess their project. Another example is the Real World Lab course at the Georgia Institute of Technology [18], where undergraduate students are involved with real industry projects and take part in performing a CMM assessment on local industry by interviews.

In order to offer an environment in which students can have hands-on experiences, most of these experiences are based on an industry partnership in which students participate in the companies' SPI projects. Yet, often software organizations are reluctant to share their quality and process issues with students and/or do not have the capacity to assign staff to those activities [8] and, therefore, such a partnership may not always be possible. In this context, an alternative for providing an environment in which students can learn to apply SPI concepts may be the combination of courses in a cross-course design. In other SE areas, the adoption of such a cross-course design has shown positive results [19, 20, 21]. Cross-course designs seem to be especially indicated when using the advantage of capstone projects being executed, which allows to apply SE concepts on larger and more complex projects within the time and resource restriction of each of the individual courses. In addition, they also can offer a more stimulating environment for teaching relevant skills, such as, communication and help to motivate SE better. Yet, so far, no experiences on such instructional designs for teaching SPI have been encountered.

3 Proposal for a Cross-Course SPI Course

One of the main research areas at the Master Program in Applied Computer Science at the UNIVALI is SPI and SPA. Therefore, master students need to acquire knowledge and skills relevant to SPEs. Students who enter the course are Bachelors in Computer Science, with basic SE knowledge and practical software development experiences. They attend at least two SE related courses – one providing a general overview on SE and one focusing, specifically, on SPI. The SPI course is offered with 4-hour lectures during 15 weeks. The objective of the course is to teach basic knowledge on SPI, mainly on defining and documenting software processes, assessing software process capability and/or maturity as well as on selected SE topics, such as, project management, on the cognitive levels of remembering, understanding and applying in accordance to Bloom's revised taxonomy [12] as well as skills, such as, communication, team work, leadership, and problem solving.

In order to achieve these objectives, we propose an educational strategy based on the constructivist learning theory through the integration of practical course work within a simulated software project being run in parallel as part of an undergraduate capstone project at the UNIVALI. Within this capstone project, undergraduate students work in teams to plan and monitor, analyze, design, implement and test a software system. This cross-course design enables the students of the SPI course to assess and define a software process for an authentic environment. For a better understanding, first the design of the graduate SPI course is explained in detail and, then, a summary of the undergraduate capstone project discipline is provided.

3.1 SPI course design

The learning objective of the graduate SPI course is that students remember and understand software process improvement and assessment concepts, models and approaches and acquire the competency to apply them with assistance in practice. The students should also reinforce their knowledge on project management and SE in general. Table 1 summarizes the lecture plan.

Table 1. Lecture plan

Unit	**No. of hours**		**Contents**	**Teaching method**	**Evaluation**
	theory	practice			
0	4	0	Course presentation (and pre-test)	- Expositive lecture with discussion - Multiple-choice exam	
1	2	0	SPI- basic concepts and approaches (IDEAL, ISO/IEC 15504-4)	- Expositive lecture with discussion	Questions in final exam
2	2	0	Software process reference models (CMMI, ISO/IEC 12207, ISO/IEC 15504-5, MPS.BR)	- Expositive lecture with discussion	Questions in final exam
3	2	16	SPA – concepts and process (based on SCAMPI and	- Expositive lecture with discussion	- Work project 1: Assessment project

			ISO/IEC 15504)	- Assessment project	- Questions in final exam
4	4	12	Software process definition and documentation -concepts and process	- Expositive lecture with discussion - Process modeling project	- Work product 2: Process definition project - Questions in final exam
5	4	0	Course debriefing (and post-test)	- Discussion - Multiple-choice exam - Lecture evaluation questionnaire	

The educational strategy of the course is based on a constructivist approach providing a hands-on experience to the students to enable them to learn how to apply those concepts and approaches in practice. Expositive lectures are reduced to a minimum, just to provide an introduction and a general overview on SPI concepts as a basis for the work projects to be done during the course. The main focus of the course is on two practical work projects to be done in groups. These work projects simulate authentic SPI situations for students to learn and exercise the application of SPA and process definition. The work projects take place in the software project being run as part of an undergraduate capstone project (see section 3.2). As input to the work projects, students receive a detailed instruction by the teacher and a set of relevant material (including, context descriptions, e.g., of the software organization and its process used in the capstone project, a definition of a SPA method and background material, such as, the CMMI model, etc.). This information as well as the artifacts being created by the students is managed on a google site (Figure 1).

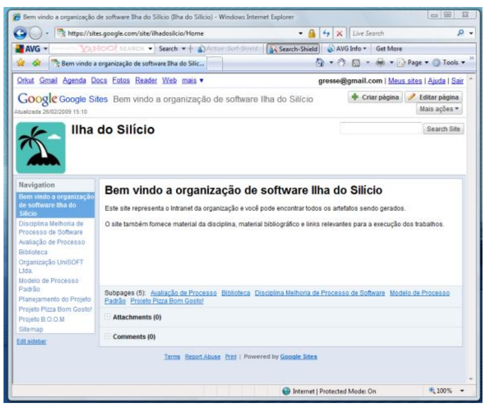
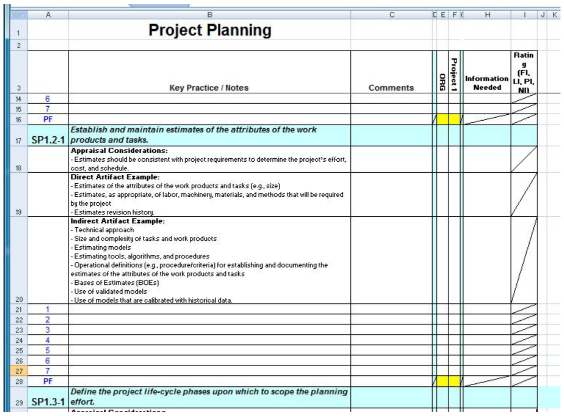


Figure 1. Google site overview of the SPI course

A large part of the work projects is done in the classroom in order to allow the students to meet as a group as well as to allow the teacher to function as a tutor, who keeps active learning going, answers questions, indicates additional material and encourages reflections. In the end of each of the work projects, the student groups present the results in the classroom and discuss and compare their results and experiences with their colleagues.

Work project 1: Process assessment. The objective of the assessment project is to teach the remembering and understanding of SPA concepts and approaches and the application of SPA in practice. As a basis for the realization of the assessment a simplified version of the SCAMPI method [22] has been predefined, including a high-level process description and document templates. During this exercise, the student groups plan, execute and analyze a SPA focusing on capability level 2 of the Project Planning (PP) process in accordance to the CMMI-DEV v1.2 model [23]. Here we focus the assessment on the Project Planning process area, as this is the first step of the capstone project in the undergraduate course, and, therefore, results of this step are available in time for the realization of the assessment. The assessment is realized in a cross-course way in the organization simulated in the undergraduate capstone project course. The students plan the assessment based on a characterization of the organization and the competencies and roles assigned to the group members of the capstone project. As further input to the assessment, the project plan produced in the capstone project is provided. Based on this information, the assessment group initiates the identification and documentation of direct and indirect evidences. All evidences collected during the assessment are documented in an EXCEL sheet (Figure 2), indicating the evidences for each of the specific and generic practices of the selected process area.



Project Planning

	A	Key Practice / Notes	Comments	ORG	Project 1	Information Needed	Rating (FI, LI, PI, NI)
14	6						
15	7						
16	PF						
17	SP1.2-1	*Establish and maintain estimates of the attributes of the work products and tasks.*					
18		**Appraisal Considerations:** - Estimates should be consistent with project requirements to determine the project's effort, cost, and schedule.					
19		**Direct Artifact Example:** - Estimates of the attributes of the work products and tasks (e.g., size) - Estimates, as appropriate, of labor, machinery, materials, and methods that will be required by the project - Estimates revision history.					
20		**Indirect Artifact Example:** - Technical approach - Size and complexity of tasks and work products - Estimating models - Estimating tools, algorithms, and procedures - Operational definitions (e.g., procedure/criteria) for establishing and documenting the estimates of the attributes of the work products and tasks - Bases of Estimates (BOEs) - Use of validated models - Use of models that are calibrated with historical data.					
21	1						
22	2						
23	3						
24	4						
25	5						
26	6						
27	7						
28	PF						
29	SP1.3-1	*Define the project life-cycle phases upon which to scope the planning effort.*					

Figure 2. Example extract of the evidence collection sheet

Then, following the assessment plan, the assessment team realizes interviews with members of the capstone project in order to collect also affirmations from the project members (Figure 3).

Based on the obtained affirmations, the data collection is completed and the obtained information is analyzed. As result of this activity, each assessment group prepares and presents a report on the assessment results indicating improvement opportunities. The assessment results are also provided as feedback to the students of the capstone project. The cross-course integration of this work project is detailed in Table 2.

Table 2. Cross-course integration in the assessment work project

Graduate SPI course		Undergraduate capstone project course
Plan for assessment	<-	Characterization of "organization"
	<-	Characterization of member's competencies and roles assignment
Obtain objective evidence and document in data collection plan	<-	Project plan
Examine objective evidence		
Realize interviews and examine evidence from interviews	<->	Interviews with members of the project team
Generate appraisal results	->	

Figure 3. Assessment interview

Work project 2: Software process definition. Based on the results obtained during the assessment, each group of the SPI course defines a software process in the context of the organization of the capstone project, in alignment with the objectives and practices required with respect to capability level 1 of the CMMI-DEV v1.2 for the Project Planning process. The process definition is done in a hybrid way, combining descriptive definition (based on the elicited informal process being executed in the capstone project) and prescriptive definition (improving identified weaknesses with respect to the practices as required by CMMI). The students document the process, describing objectives, activities, methods, techniques and tools to be used as well as work products to be consumed and generated (including, the definition of templates for all work products to be generated) and the identification of roles and responsibilities. The process definitions are documented in a demo version of the Enterprise Architect tool (http://www.sparxsystems.com.au/) (Figure 4). In addition, the students explicitly track the compliance of the defined process to all required specific practices of the PP process of the CMMI-DEV v1.2.

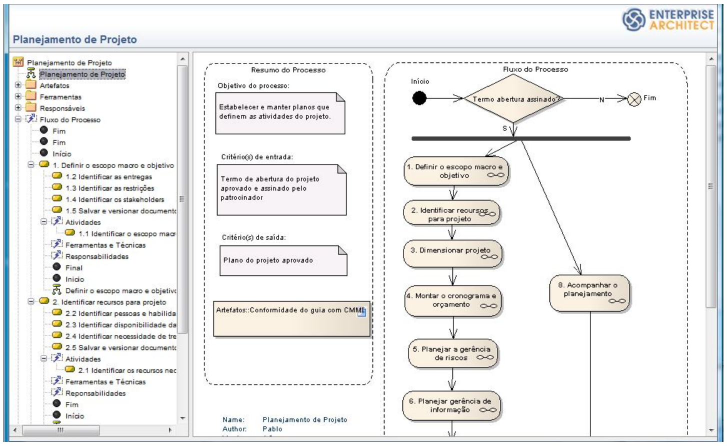


Figure 4. Example extract of a developed process definition

Evaluation: The achievement of the expected learning outcomes is assessed based on the evaluation of both work projects (SPA and process definition) and the result of an exam in the end of the course. The work projects are evaluated with regard to completeness, correctness, clarity, consistency of the produced results and the organization and knowledge shown during the presentation. The exam is a multiple choice test including questions on the cognitive levels of remembering, understanding and application.

3.2 Capstone Project Course

At the Computer Science undergraduate program at the UNIVALI, software engineering concepts are introduced in two software engineering courses (SE 1 and SE 2) covering basic concepts and the software process focusing on requirements development, design, testing, project and quality management. Two subsequent courses (APS1 and APS2), focus on the application of this theoretical knowledge and provide “hands-on” experience. In the APS2 course, students realize a semester-long capstone project in groups of 4 to 6 students. They plan and execute a software project executing requirements analysis, design, implementation and testing.

The project starts with a planning phase, in which the students plan the project using a pre-defined project plan template. Then, they start the technical activities following a predefined waterfall process model consisting of 4 phases: requirements development, design, implementation and testing. During the execution of the project the students collect data (basically, on the effort spent and start and end dates of activities) and in the end a project post-mortem is realized. To enable close accompanying of the student work and the provision of early feedback, each team has to deliver and present its results at the completion of each phase of the project.

The role of the customer is played by the teacher. Each student is assigned to a specific role (e.g., project manager, requirement analyst) indicating his/her primary responsibility. Yet, due to the learning objectives, all students participate in all phases of the capstone project.

4 Preliminary Evaluation

As part of the change of the course design, a preliminary evaluation of the new design has been performed in the first semester of 2009. The objectives of the evaluation are to analyze:

O1. If a positive learning effect on the cognitive levels of remembering, understanding and applying level and/or skills can be observed;

O2. If the course design is considered appropriate in terms of teaching method, adequacy of work project, cross-course integration and utility in practice; and

O3. What are the course strengths and weaknesses?

The objective of this preliminary evaluation is rather to obtain a first subjective evaluation of these aspects from the student's point of view.

These research questions have been analyzed based on the Kirkpatrick's four-level model for evaluation [24], a popular and widely used model for the evaluation of training and learning. In accordance to Kirkpatrick's four-level model for evaluation, we investigate all objectives on level one: reacting, which focuses on how the participants feel about the learning experience by collecting data via satisfaction questionnaires. We investigate objective 1 also on level two: learning, which focuses on the evaluation of the increase in knowledge by administering a pre-and post-test. On level 2, we evaluate the learning effect separately for each of the knowledge levels (remembering, understanding, applying) by comparing the average scores between pre-test and post-test (relative learning effect).

Different kinds of data were collected, including the realization of a pre-test exam in the beginning of the course as benchmark and a post-test exam in the end of the course. Both exams were multiple choice tests with similar content and degree of difficulty. In addition, subjective data has been collected via questionnaire from the students in the end of the course.

4.1 Results

The proposed course design in the SPI course has been applied during the 1. Semester 2009 at the Master Program on Applied Computer Science at the UNIVALI. In total, 5 students attended the course. In general, we obtained a very positive feedback with respect to the new course design.

O1. Can a positive learning effect on the cognitive levels of remembering, understanding and applying level and/or skills be observed?

This question has been analyzed by comparing the results in the pre-test and post-test. In general, the average difference is 17.2 points (with a total number of 80 points per test) varying from a difference of 7 to 33 points, indicating that the knowledge of the students increased. It can also be observed that the greatest knowledge increase took place on the cognitive level of application (Figure 5), as intended in the learning objective of the course.

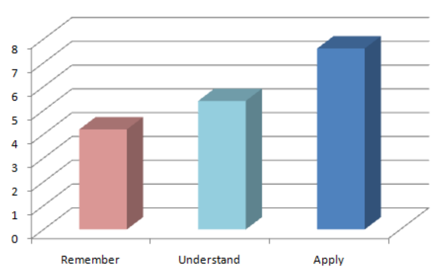


Figure 5. Average difference of knowledge per cognitive level

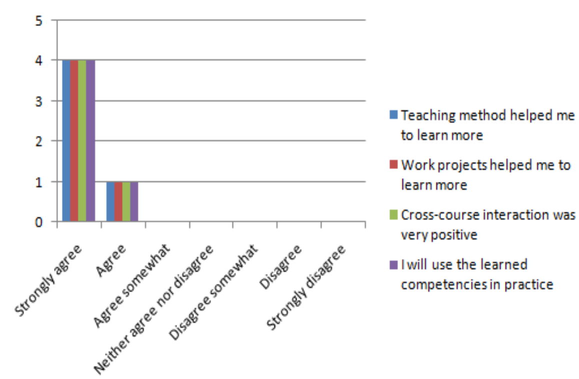


Figure 6. Distribution of number of student responses

The subjective evaluation by the students also indicated that the students believe that the course helped them to evolve relevant skills, such as, team work, problem solving, communication, etc.

O2. Is the course design considered appropriate in terms of teaching method, adequacy of work project, cross-course integration and utility in practice?

These aspects have been analyzed based on subjective data collected from the students at the end of the course. On a 7-point likert scale, all students either agreed or strongly agreed that the teaching method, the work projects and the cross-course integration are appropriate and they pretend to use the acquired competencies in practice (Figure 6).

O3. What are the course strengths and weaknesses?

Based on the feedback obtained by the students, the principal strengths of the course design are:

- Presentation of theoretical concepts constantly in relation with practical application;
- Strong emphasis on practical work;
- Practical work in combination and interaction with the undergraduate course;
- Presentation and discussion of the results of the practical work projects in the classroom; and
- Organization of the two practical work projects as a sequence (second work project building upon the results of the first one).

As principal weaknesses the students cited:

- Small number of students attending the course and, consequently, the formation of only very small groups;
- Lack of a complete working example; and
- Ineffective usage of the time reserved for practical work in classroom.

4.2 Discussion

Although, the results of this preliminary evaluation have to be interpreted with extreme caution due to the very small sample size and its restriction to only one application, the results may provide a first indication that the new course design has a positive impact on the learning effectiveness. It seems that the constructivist approach, in which learning is defined as an active process for knowledge building rather than a knowledge acquisition process, contributes positively to the learning of knowledge on the application level as well as relevant skills. Yet, we also observed that just providing a learning environment, literature and a general introduction may not be sufficient. Students (maybe, due to the fact, that they are more used to traditional classroom teaching) expect a more guided approach. We therefore, intend to include more expositive lectures substituting independent literature study by the students on their own. Another alternative is also the integration of more diverse teaching methods, including, for example, games and case studies.

The cross-course design of the course was considered a very positive aspect of the course ensuring a richer learning experience. The students of both, the graduate and the undergraduate course, liked the experience very much. It turned the assessment into a “real” experience applied to a project and people outside their own course, which was executed with great care and in a professional way.

Another issue emphasized by the students was the presentation and discussion of the results of the work projects in the classroom. The students expressed that these offered them an additional opportunity for learning by examples, especially, as no “golden solution” for the work projects was available.

The course design also seems to be able to deal with varying levels of background, as within this application, the students background varied from professionals with PMP certification to students who finished graduation ten years ago and were just starting to learn about software engineering.

5 Conclusions

Teaching the application of SPI concepts is challenging. In this paper, a cross-course design is proposed combining a graduate SPI course and an undergraduate capstone project in order to recreate an authentic environment, which allows students to acquire practical knowledge and experience. First experiences in applying this design provide a preliminary indication for a positive impact on learning. Implementing the improvement opportunities identified, we intend to repeat the application of this course design in the SPI course, collecting also feedback on a larger scale.

Acknowledgements

We would like to thank the students of the SPI and the APS2 course at the UNIVALI during the 1. Semester 2009. This work was supported by the CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*) and CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*), entities of the Brazilian government focused on scientific and technological development.

Literature

1. SEI. Process Maturity Profile CMMI for Development SCAMPI Class A Appraisal Results 2008 End-Year Update. Software Engineering Institute, Pittsburgh, March 2009.
2. IISP. Software Process Improvement Body of Knowledge (SPIBOK) http://www.spinstitute.org/spibok.htm
3. J. S. Hawker. A Software Process Engineering Course. American Society for Engineering Education. Proceedings of the 2009 American Society for Engineering Education Annual Conference, Austin, TX, 2009.
4. R. McFeeley. IDEAL: A User's Guide for Software Process Improvement. Handbook CMU/SEI-96-HB-001, Carnegie Mellon University/Software Engineering Institute, Pittsburgh, 1996.
5. IEEE Computer Society. SWEBOK – Guide to the Software Engineering Body of Knowledge, 2004 version, IEEE Computer Society, 2004.
6. The Joint Task Force on Computing Curricula IEEE CS/ACM. Software Engineering 2004 - Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, 2004.
7. E. Biberoglu & H. Haddad. Survey of Industrial Experiences with CMM and the Teaching of CMM Practices. Journal of Computing Sciences in Colleges, vol. 18, no. 2, pp 143 – 152, December 2002.
8. M. L. Jaccheri. Software Quality and Software Process Improvement Course based on Interaction with the Local Software Industry. Computer Applications in Engineering Education, John Wiley & Sons, Volume 9, Issue 4, pp 265 – 272, Mar 2002.
9. F. Percival et al. Handbook of Educational Technology, 3. ed. Kogan Page, London, 1993.
10. J. Choi, M. Hannafin. Situated Cognition and Learning Environments: Roles, Structures and Implications for Design. Educational Technology Research Development, vol. 43, no. 2, pp. 53–69, 1995.
11. M. L. Jaccheri & P. Lago. How Project-based Courses face the Challenge of educating Software Engineers. Proc. of the Joint World Multiconference on Systemics, Cybernetics and Informatics (SCI'98) and the 4th International Conference on Information Systems Analysis and Synthesis, Orlando, USA, pp. 377-385, 1998.
12. L. W. Anderson, D. R. Krathwohl (Eds.). A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. New York: Longman, 2001.
13. S. Hadjerrouit. Toward a Constructivist Approach to Web-based Instruction in Software Engineering. Proc. of Norwegian Computer Science Conference. Oslo/Norway, November 2003.
14. E. Bareiša et al. Software Engineering Process and its Improvement in the Academy. Information Technology and Control, Vol.34, No.1, 2005.
15. T. Dingsøyr et al. Teaching Software Process Improvement through a Case Study. Computer Applications in Engineering Education Computer Applications in Engineering Education, Volume 8, Issue 3-4, pp. 229 – 234, Nov 2000.
16. G. W. Hislop. Teaching Process Improvement in a Graduate Software Engineering Course. Proc. of 29th ASEWIEEE Frontiers in Education Conference, San Juan/Puerto Rico, 1999.
17. P. N. Robillard, J. Mayrand, J.-N. Drouin. Process Self-Assessment in an Educational Context. Software Engineering Education, Springer Lecture Notes in Computer Science, vol. 750/1994, pp. 211-225, 1994.

18. M. M. Moore & T. Brennan. Process Improvement in the Classroom. Software Engineering Education, Springer Lecture Notes on Computer Science, vol. 895/1995, pp. 123-130, 1995.

19. C. Brown & R. Pastel. Combining Distinct Graduate and Undergraduate HCI Courses: an Experiential and Interactive Approach. Proc. of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga/USA. pp. 392-396, 2009.

20. G. Sindre et al. The Cross-Course Software Engineering Project at the NTNU: Four Years of Experience. Proc. of the 16th Conference on Software Engineering Education and Training, IEEE Computer Society, Washington/USA, 2003.

21. R. J. Fornaro et al. Cross-functional Teams Used in Computer Science Senior Design Capstone Courses. 30th Annual Frontiers in Education Conference, 2000.

22. SCAMPI Upgrade Team. Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.2: Method Definition Document. Handbook CMU/SEI-2006-HB-002. Carnegie Mellon University/ Software Engineering Institute, Pittsburg, 2006.

23. CMMI Product Team. CMMI for Development, Version 1.2. Technical Report CMU/SEI-2006-TR-008, Carnegie Mellon University/ Software Engineering Institute, Pittsburg, 2006.

24. D. L. Kirkpatrick, J. D. Kirkpatrick. Evaluating Training Programs: The Four Levels, 3rd edn. Berrett-Koehler Publishers, San Francisco, pp 379, 2006.

Author CVs

Christiane Gresse von Wangenheim

Christiane Gresse von Wangenheim is a professor at the Federal University of Santa Catarina (UFSC). Her research interests are software process improvement, including project management. Previously, she worked at the Fraunhofer Institute for Experimental Software Engineering and the UNIVALI. She received a PhD in Production Engineering at the Federal University of Santa Catarina (Brazil) and a PhD in Computer Science at the University of Kaiserslautern (Germany). She's also a PMP - Project Management Professional and Assessor of the Brazilian Process Improvement Model MPS.BR. She's a member of the IEEE Computer Society, the Project Management Institute, and the Working Group ISO/IEC JTC1/SC7/WG24—SE Life-Cycle Profiles for Very Small Enterprises. Contact her at UFSC, Campus Universitário 88049-200 Florianópolis/SC, Brazil; gresse@gmail.com

Jean Carlo R. Hauck

Jean Carlo Rossa Hauck is visitor researcher at Dundalk Institute of Technology (DkIT) and SEPG manager of the CYCLOPS Research Group at the Federal University of Santa Catarina (UFSC). His research interests are in software process improvement and project management. He received his M.Sc. in Computer Science from the UFSC and he is PhD student of Knowledge Engineering and Management at the Federal University of Santa Catarina. Contact him at UFSC - EGC, Campus Universitário 88049-200 Florianópolis/SC, Brazil; jeanhauck@gmail.com