Volume 24 Number 4 2008 ISSN 0949-149X

The International Journal of Engineering Education

Contents

Part I
Special Issue: Trends in Software Engineering Education
Guest Editors: Levent Yilmaz and Stephen H. Edwards

In this paper, we present a strategy to help students develop the necessary skills to become effective software engineering professionals. We created a software engineering group, called QualDev, composed mostly of undergraduate students. QualDev places students on real software projects, but with some features to ease their control and evaluation. Our educational strategy is to use active teaching/learning methodologies that enable us to create scenarios with regular self-assessment. There are many challenges related to setting up and maintaining such a software development team; we relate our experience in creating and evolving the QualDev group, its organization, projects, methodologies, and processes.

Keywords: software engineering education; team work; active learning; collaborative learning

The software industry uses a mixture of plan-driven and agile techniques, and educators must prepare students for industry needs while creating an effective educational environment that appeals to a diverse student population. We describe the undergraduate course in software engineering at North Carolina State University, which teaches both agile and plan-driven practices while emphasizing collaborative and active learning. We present demographics, personality types, and learning styles from 400 students, and provide statistical analyses and student testimonials on the impact of our course. Students have reacted favorably to the course and are better prepared to meet the diverse needs of industry.

Keywords: software engineering education; agile methods; personality types; learning styles

Project courses are an important component of some software engineering curricula. They are capstone projects where teams of students experience the various practices for developing software. Instructors play the roles of coaches in guiding the students during the various phases of their project. Nowadays, software development processes fall into two major paradigms. The Disciplined software process paradigm defines best practices and their relationships on the basis of roles, activities and artifacts. The Agile process paradigm, which is based on values of simplicity, communication, and feedback, uses simple practices to enable a team to tune the practices to their unique situation. The two process paradigms have great value in general and one is likely to be more efficient than the other in any specific development project. However, it could be interesting to find out how each of these process paradigms performs in learning environments. To achieve this we conducted an observational study in an academic environment. Six teams of four students developed their own versions of a software product based on the same requirements. Three teams used a Disciplined process and three teams used an Agile process. This study is based on four observations: the quality of the implementation of the requirement, the total project effort, the process activity effort and the product size. The data to support each of these observations are presented. In this study, however, the Disciplined paradigm provides less project implementation with a better realization of quality. This study indicates that the more efficient approach for capstone projects for inexperienced students in software engineering would be a Disciplined process paradigm.

Keywords: software engineering; software process; disciplined process; agile process; capstone project

Research in software engineering education has, in recent years, attempted to achieve the equilibrium between academia and practice. The software engineering education research community has obtained a number of valuable outcomes in the areas of content curriculum, pedagogy, and technology, respectively. However, very few studies have successfully integrated these three dimensions into a single learning environment. This study developed and evaluated a Game-Based Software Engineering Educational System (GBSEES) for software engineering education. GBSEES adopted a role-playing strategy using a digital game-based learning model. This game-based system was based on the educational theory of Technological Pedagogical Content Knowledge, which integrates pedagogical knowledge, content knowledge, and technological knowledge. In the game-based learning system, students learned about the process of software development in a team-based environment by using a role-playing gaming strategy. The study also investigated the effect of the GBSEES on the students' attitude to learning.

Keywords: software engineering education; technological pedagogical content knowledge; digital game-based learning; system development and evaluation

S. Minocha, M. Petre and **D. Roberts** 689–704 Using WIKIS to Simulate Distributed Requirements Development in a Software Engineering Course

Software development activities are increasingly being conducted collaboratively across multiple time zones and multiple teams. This creates challenges in building shared values and trust, and in coping with asynchronous collaboration and communication. In response to these trends, tools such as wikis, blogs, web portals and groupware are being integrated in development processes to enhance the productivity and effectiveness of teams. To enable students to meet these challenges, there is a need to use technology in software engineering education to simulate authentic structures of work practices. Use of collaborative and discourse tools will provide students with the experiences of communicating and negotiating with diverse stakeholders with different views and backgrounds. It will also enable the development of transferable skills for working with community tools in the industry. As with most software design and development processes, Requirements Engineering (RE) is increasingly being conducted in distributed environments. Wikis are being used to provide a platform for asynchronous collaboration for participative requirements development. In a post-graduate RE part-time distance-learning course at the Open University in the UK, we have introduced wiki activities in the course to provide students with the opportunity to engage in small-group collaboration to emulate RE practice. In this paper, we discuss the nature of the RE process, the usage of wikis in RE practice, and the challenges of introducing collaborative-work and wikis on the RE course at the Open University and our solutions. We will draw on empirical evidence to discuss effectiveness of wiki in collaborative learning of the RE processes.

Keywords: wiki; collaborative learning; requirements engineering; software engineering education; virtual teams

K. A. Gary 705–716 The Software Enterprise: Practicing Best Practices in Software Engineering Education

Software engineering educators emphasize teaching concepts in software engineering principles and then applying them in the context of a capstone project. Capstone experiences often focus on leveraging a popular process model. The emphasis on process provides a structure for coordinating team activity, with an objective of demonstrating to the student the value of following a process model. We contend that more emphasis is required on detailed process execution than is given proper due. Specifically, best practices are now emphasized in the software engineering profession over rigid process structures, and as educators we must respond to this cultural shift by teaching the role of best practices in a broader applied process context. Our approach in the Software Enterprise, our multi-year capstone sequence at Arizona State University Polytechnic, is to provide a process structure, teach best practices, and then give teams 'just enough rope' to resolve issues by leveraging the process, best practices, and soft skills. The Software Enterprise presents a unique, iterative accelerator for presenting software engineering from concepts through to applied practice. This pedagogical model allows us to present, practice, and apply best practices in the context of real scalable projects, resulting in better contextual learning for our students. In this paper we describe the machinery for teaching software engineering in this manner and present some preliminary survey results evaluating how well Enterprise students apply these skills in practice.

Keywords: engineering education; software process; software engineering capstone; software enterprise

G. Jimenez-Diaz, M. Gomez-Albarran and P. A. Gonzalez-Calero 717–728 Teaching GoF Design Patterns through Refactoring and Role-Play

In order to fully understand the implications of object-oriented design patterns, students need to consider alternative designs to a problem and to analyse these solutions in terms of coupling, cohesion and extensibility. Lecture-based approaches to teaching design patterns do not provide students with the insights needed unless they already have experience in object-oriented design. In this paper we present an approach to teaching design patterns that promotes active learning and makes students participate in refactorings through role-play sessions. We describe two experiments that demonstrate student acceptance and present promising results on the effectiveness of the approach.

Keywords: pattern-directed refactoring; active learning; role-play; object-oriented design pattern learning

D. A. Carrington 729–737 What can Software Engineering Students Learn from Studying Open Source Software?

There is a large gap between the scale and complexity of typical software products and examples used in software engineering education. Since complexity is considered an essential property of software systems, this gap creates a problem for software engineering students and educators. Studying open source software can provide software engineering students with realistic and challenging examples and pragmatic instances of abstract concepts such as software design patterns. For software engineering educators, the vast array of freely available software sources allows selection to suit their educational objectives and constraints. This paper reviews how open source software is used in a software engineering studio course and discusses the outcomes from the perspectives of students and educators.

Keywords: engineering education; open source community; learning environments; distributed software development

N. Petalidis 738–746 Adopting Lakatos in a Software Engineering Course

The standard practice in a software engineering course is to present the theory as a list of dogmatic guidelines. In this setting problems appear artificial and consequently students fail to appreciate them. Similarly, solutions arrive magically, letting students believe that this is the norm. The value of an incremental and iterative methodology is therefore missed. A different approach, borrowed from Lakatos [1], is presented here. Students are given a problem and through 'trial-and-error' discover their own solutions. Unlike a typical term-project that follows the theory, it is the problem that drives the theory. The result is better appreciation and comprehension of software engineering notions.

Keywords: software engineering education; constructivism; reflective learning and teaching

P. Lago, H. Muccini, L. Beus-Dukic, I. Crnkovic and **S. Punnekkat** 747–760 GSEEM: a European Master Program on Global Software Engineering

This paper presents a novel European Master programme on Software Engineering (SE), being put forward by four leading institutions from Sweden, UK, Netherlands and Italy. The Global SE European Master (GSEEM) programme aims to provide students with an excellence in SE based on sound theoretical foundations and practical experience, as well as preparing them to participate in the global development of complex and large software systems. GSEEM has been designed with two aspects of note: 1) the three specialization profiles in which the consortium excels: Software Architecting, Real-time Embedded Systems Engineering, and Web Systems and Services Engineering; 2) an innovative concept of 'shared modules', delivered all together by multiple institutions. Four types of shared modules are foreseen: 'parallel' twin modules, which run remotely between universities, 'shifted' modules, which teach SE concepts incrementally with shifts in study locations and timeline, 'complementary' modules in which complementary SE concepts are taught in parallel through shared projects, and 'common' modules, which share the presentations and the project. The profiles realize 'integrated knowledge' by complementing partial knowledge available at partner institutions. The paper presents some of the important issues faced during the design of the program and explains how GSEEM achieves the objectives of educating global software engineers. The lessons learned from the GSEEM design are of a technical, pedagogical and organisational or administrative nature.

Keywords: European Master Program; teaching Global Software Engineering; shared modules

Empirical studies in software engineering are essential for the validation of different methods, techniques, tools, etc. Students play a fundamental role in carrying these studies out successfully and, as a consequence, most experiments connected with software engineering are conducted in academia. Benefits which are concerned exclusively with aspects of research are not the only ones to come from studies of this kind: it is very important also to consider benefits from a teaching point of view. Therefore, when experiments are conducted in academia, they must be planned not only to obtain insights into research but also to help students who participate as experimental subjects.

Keywords: computer science education; software engineering; student experiments

Part II
Contributions in: Engineering Education Research, Civil Engineering, Electronic Engineering, Simulation and Manufacturing Engineering

Recent years have seen an increase in transformations in educational methods towards Problem Based Learning (PBL). In the process of organizational change, staff development remains one of the key elements. This paper presents a pioneer program in staff development based on PBL learning principles, the Masters in Problem Based Learning in Engineering and Science (MPBL) at Aalborg University, Denmark. Drawing on current experiences and reflection, the paper discusses the outcomes as well as the existing challenges in combining a PBL approach with technology-supported on-line delivery as a strategy for staff development in engineering education.

Keywords: engineering education; staff development; problem based learning; e-learning.

In engineering education, one-on-one tutoring is commonly used to stimulate learning. This study develops and refines a practical tutoring model. The model is based on naturalistic tutoring augmented with elements from cognitive theory that are specifically missing in naturalistic tutoring. A four-week case study of near-peer tutors in engineering dynamics suggested four findings. First, interactions should be based within the tutee's cognitive framework. Second, deep exploring of the tutee's pre-existing knowledge leads to tutoring within the tutee's cognitive framework. Third, four tutoring actions emerged that address learning needs while keeping the tutoring within the tutee's framework. These tutoring actions are: a) guided assistance in problem solving, b) prompting tutees to construct and reconstruct what they know, c) explicit problem structuring and, d) presenting new information only when needed. Fourth, deep exploring facilitates the development of strong rapport. The paper concludes with a presentation of a revised tutoring model.

Keywords: tutor; tutoring model; cognitive; naturalistic tutoring; engineering

Stress is any change that requires a person to adjust to a new situation and is registered in the brain as a threat that causes a stress response in the body. 'Student tardiness' is the term ascribed to students who suffer from tardiness, which is defined as the quality or condition of not being on time. Student tardiness is currently experienced in many higher educational institutions, contributing to poor academic achievements and subsequent low throughput rates. Tardiness is viewed as a construct, which is an abstract concept deliberately created to represent a collection of concrete forms of behaviour including stress or anxiety. The case study on which this article is based explored the relationship between stress or anxiety and tardiness of engineering students, establishing a direct correlation between stress or anxiety and the throughput rate of an engineering module called Design Project III. The empirical study incorporated an ex-post facto study involving a pre-experimentallexploratory design using descriptive statistics. The results of this research were applied to three separate tests which indicated a statistically significant relationship between stress or anxiety and the final throughput rate (75%) of the module Design Project III. Moreover, the presence of a negative correlation indicates that a decrease in the negative aspects of stress or anxiety will result in an increase in the final throughput rate of the module, subsequently influencing the academic success of engineering students.

Keywords: student tardiness; throughput rate; engineering student; stress; anxiety

Accreditation for off-campus engineering programmes has proven to be problematic. In Australia, off-campus programmes are compelled to contain mandatory residential sessions so that off-campus students can have an 'on-campus experience'. This paper explores the nature of modern on-campus undergraduate engineering study, and finds that it now typically involves at least part-time employment and has more in common with off-campus study than the on-campus experience enjoyed by most of the current institutional (education and professional) administrators when they completed their undergraduate studies. Rather than ignore student term-time work, engineering programmes should use it to enhance the development of desirable graduate attributes.

Keywords: student attendance; student employment; study and work

We present and explore factors which mainly affect the satisfaction level of engineering students in Pakistan. Our questionnaire is based on student expectations from their institutions. Randomly collected data from 225 students are analysed using software MINITAB 14, six-sigma techniques of Measurement System Analysis, Affinity diagram, Pareto Analysis, SIPOC analysis, Cause and Effect matrix and Scatter plots. We find that the teaching skill of teachers is the most critical factor. This work can guide the educational leadership in focusing their resources for best satisfaction of their students.

Keywords: Engineering students, satisfaction, teaching skills, six-sigma.

Y. K. Hoh 817–824 Presenting Female Role Models in Civil Engineering: An Outreach Activity to Help Teachers Overcome Their Misperceptions of Engineers

This paper describes an activity the author has carried out with 72 high school science teachers to enable them to overcome their misperceptions of engineers and engineering. The activity introduced them to prominent women in civil engineering, and raised their awareness of these female engineers' contributions to engineering and society. The results showed that the activity was effective in dispelling the teachers' misperceptions. The female civil engineers featured in this activity cited the role of their parents or teachers in encouraging their pursuit of an engineering career. They held senior positions in academia, government or industry. They acknowledged that they had encountered difficulties at their workplaces but they also mentioned progress made towards acceptance and equality. Teachers and professors can use the examples of these prominent female engineers as role models to inspire their female students who are aspiring to become civil engineers.

Keywords: Civil engineering; engineers; outreach; role models; women; workplace diversity

A. Perles, J. Albaladejo, J. Vte. Capella, J. M. Martinez, H. Hassan and **C. Dominguez** 825–832 Design and Application of a Data Acquisition Card Simulator for Electronic Engineering Studies

Active teaching methods in large groups of students have been applied using a simulator. This approach also allows students' work to be controlled outside the classroom, at a moderate laboratory cost. The tool is a data acquisition card simulator for the development and testing of computer applications in industrial control. It works by simulating the electrical signals connected to its pins and the physical behaviour of the processes that it controls.

Keywords: computer science for engineers; data acquisition card; simulation software

J. M. P. Cardoso 833–842 A Teaching Strategy for Developing Application specific Architectures for FPGAs

This paper presents an approach to teaching design of non-programmable application-specific architectures using VHDL, logic and physical synthesis tools and FPGAs. The approach relies on mini-projects that resemble typical problems that students may face in real-life concerning the design of application-specific architectures. The teaching approach presented in this paper supports the incremental learning of both VHDL and the tools used, as the projects are being developed, i.e., students are motivated to acquire skills at the pace at which those skills are required to advance project development. The results so far are very encouraging. Even students with little knowledge of hardware design and embedded systems have succeeded in their assignments. Feedback obtained from students reveals the suitability of certain aspects of the approach and the major difficulties they have faced.

Keywords: FPGAs; VHDL; application-specific architectures; digital systems; education

Y.-B. Park, Y. Lee, J. Kang and **B. Wang** 843–849 The Effects of 3D-Simulation-Based Instruction on Students' Achievement and Interests in a Manufacturing Engineering Class

The purpose of the study was to compare and analyze the effects of two instructional methods—instructor-led and simulation-based instructions—on engineering students' achievements and course interests in a manufacturing engineering class. Twenty-nine undergraduate students participated in the study, and repeated measures were employed to collect multiple sets of data. The study showed no significant differences in the means of achievement and interests. The results are discussed in conjunction with the data tables.

Keywords: manufacturing; 3D simulation; course interest