Computer Science Education
2002, Vol. 12, No. 1–2, pp. 5–9

0899-3408/02/1201–2-005$16.00

Bridging Academic Software Engineering Education and Industrial Needs

Hossein Saiedian
Elec. Engineering and Computer Science, University of Kansas, Lawrence, USA

Welcome to the special issue of the *Computer Science Education* on Software Engineering Education and Training. This special issue of the *Computer Science Education* includes four of the best education papers of the 23rd International Conference on Software Engineering (ICSE) as well as one invited article presenting ongoing work in Software Engineering Education (SEE). The articles share a common theme: They emphasize effective education for practical and industrial problems. Some of the articles, in fact, represent case studies of collaborative projects between members of the academic and industrial or government institutions.

ICSE, the flagship conference of the software engineering community, is well known for being a world-wide forum where researchers, practitioners, and educators together define and identify new directions of research and practice in software engineering, assess the state of the art, and exchange ideas. The 23rd ICSE, which included a track in SEE, was held in Toronto, Canada, during May 12–19, 2001. Four of the SEE papers were recommended for further consideration for the *Computer Science Education*. The authors of the selected papers were requested to revise their original papers and thus were provided with an opportunity to expand on their papers beyond the version published in the proceedings. The invited article provides an investigation into one type of industry/university collaboration to address the existing lack of available software engineers.

Formalism and Component-Based Software Development

Component-based software development is widely acknowledged as fundamental for improving software productivity and quality. To reason about a component in a modular fashion, without being concerned about the imple-

Correspondence: Hossein Saiedian, Elec. Engineering and Computer Science, University of Kansas, Lawrence, KS 66045, USA. Tel.: +1-913-897-8515, E-mail: saiedian@eecs.ukans.edu

mentation details of the components it reuses and without the knowledge of the entire system in which the component is deployed, participating components must have abstract specifications of behavior. Traditionally, the educational community has been reluctant in introducing component-based principles of templates, specification, and reasoning in introductory undergraduate classes for one of two reasons: The principles might be too difficult for freshmen to understand or that they might displace other principles taught in introductory courses such as efficiency analysis. The article by Murali Sitaraman (Clemson University, USA), Timothy Long and Bruce Weide (Ohio State University, USA), James Harner and Liqing Wang (West Virginia University) provide significant evidence to help overcome this reluctance. It discusses a multi-year educational experiment at two major institutions in teaching component-based software development using RESOLVE, and presents positive results from attitudinal and other evaluations of students.

Educating Students in Risk Management

The increasing pace of change in information technology (IT) makes one-size-fits-all, cookbook solutions increasingly inadequate. Yet students are largely educated on cookbook solutions to set-piece problems (e.g., compiler design and development). Applying cookbook solution approaches to current IT applications frequently leads to:

- Good solutions to the wrong problems;
- Large amounts of late rework;
- Overemphasized or underemphasized activities through inability to determine how much is enough?

A good education in risk management provides skills and methods for dealing with these problems in the following ways:

1. Addressing the risks of building the wrong system focuses software engineers on understanding the stakeholders objectives and context while exploring solution approaches;
2. Resolving risks early avoids extensive late rework;
3. 'How much is enough' questions are best addressed by considering the risks of doing too much or too little.

Educating students in risk management is not easy. Usually risk-management skills take years to acquire. The major challenges are learning how to recognize and deal with particularly risky personal tendencies and external

constraints. Barry Boehm and Daniel Port (University of S. California, USA) discuss a number of such challenges in detail and use a cognitive demands analysis to determine which individual skills are more important to learn and what sequence of educational experiences are likely to be most effective in helping the students learn the skills.

Maintenance Education is Critical

Maintenance has always been viewed as a second class programming task with an "admixture of on-the-job training for beginners and low-status assignments for the outcasts and the fallen" (Gunderman, 1988). Usually less experienced programmers are assigned to maintenance programming, while the more experienced ones begin new developments. However, junior and less-experienced programmers often lack formal training in performing their chores, which include many challenging activities such as understanding the existing program, making the necessary changes or additions, conducting regression testing, and ensuring that design and analysis documents are updated to reflect the changes, additions made, or both.

A highly skilled maintainer is the most important organizational asset for achieving quality software and is strategic for improving maintenance and development processes. This requires that universities properly prepare students to enter the maintenance workforce and that maintenance organizations actively build and maintain their human knowledge and skill base. Mira Kajko-Mattsson (Stockholm University, Sweden) Stefan Forssander, Gunnar Andersson, and Ulf Olsson (ABB, Sweden) present CM3: Maintainer's Education and Training – a maturity model for educating and training maintenance engineers within corrective maintenance. This model is presently being developed at ABB by the Software Maintenance Laboratory. Our goal is to provide guidance to ABB and industrial organizations worldwide in the process of building or improving their most important asset – humane recourses.

Software Factory Courses

Industry often express concerns that current academic curricula do not address the practical issues of real software development. John Tvedt, Kevin Gary (Catholic University of America, USA) and Roseanne Tesoriero (Experimental Software Engineering, USA) outline a proposal for an innovative core curriculum for a Bachelor of Science in Computer Science. The proposed core curriculum contains elements of traditional computer science programs combined with software engineering via a team-oriented, hands-on approach

to large-scale software development. In addition to traditional lecture, project, and exam courses, students are required to take an eight-semester sequence of 'Software Factory' courses. Software Factory courses help students' put their newly acquired skills to work in a real software organization staffed and managed by all students in the program.

Lack of Skilled Software Engineers: A Concern

During the past several years, the market for skilled software engineers has been growing. Even the current slowdown in the technology sector has not allowed the supply for software engineers to meet the existing demand. The inequity between supply and demand has resulted in a variety of problems like high turnover rates, staffing shortfalls, increased production costs, increased salaries, and outflow of crucial company knowledge. In addition to the increase in demand for software engineers, some companies that have software development as a focus have indicated that the skills provided by existing software engineering education do not completely meet their requirements.

Since the most influential factor in the success of a software project is the skill level of the participating software engineers, and one of the highest risk factors for a project is the lack of talented software personnel to develop the project (Boehm et al., 2000), the expansion of the pool of skilled software engineers is a critical task. One obvious solution to the dual problems of the shortage of skilled software engineers and the software development knowledge shortfall is to involve industry more closely in the educational process. The participation of the companies that stand to benefit most from the education has several potential benefits which include students with a skillset that matches industry needs, increased company visibility, increased incentive on the part of the students, and others.

The investigation into one type of industry/university collaboration to address the existing lack of available software engineers is the focus of the paper by Heidi Ellis (Rensselaer at Hartford, USA), Nancy Mead (Software Engineering Institute, USA), Ana Moreno (Universidad Politecnica de Madrid, Spain), Cynthia Tanner (West Virginia University, USA), and Dawn Ramsey (Southern Polytechnic State University, USA). The paper looks specifically at efforts to re-educate non-software engineers to become software engineers. The research provides insight into the factors that contribute to the success of such a collaboration and outlines guidelines for the development of successful industry/university collaboration programs for producing software engineers. The authors hope that these guidelines can be used by both

companies and academic institutions alike to structure successful programs for the re-education of software engineers.

I thank the organizers of the conference, the authors, and the reviewers for all their efforts which made the production of this special issue of the *Computer Science Education* possible, and hope you enjoy their collective contributions.

REFERENCES

Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., & Steece, B. (2000). *Software cost estimates with COCOMO II*. Englewood Cliffs, NJ: Prentice Hall.

Gunderman, R. (1988). A glimpse into a program maintenance. In G. Parikh (Ed.), *Techniques of program and system maintenance*. MA, USA: Wellesley. (pp. 55–59). QED Information Sciences.