ELSEVIER

Information and Software Technology 40 (1998) 191–193

INFORMATION AND SOFTWARE TECHNOLOGY

Software engineering education: an international perspective

Hossein Saiedian

Department of Computer Science, University of Nebraska at Omaha, Omaha, NE 68182-0500, USA

1. Background

The state of software and software engineering practice today includes many successes and many great products. The field itself is thriving and in many ways contributes to our daily lives, from every day credit card uses in the grocery stores (which depend on computer-based point-of-sale systems) to the country's space mission and defense capabilities. But there are also major software failures and chronic crisis. In an excellent article which appeared in *Scientific American*, Gibbs briefly describes a representative sample of such failures and problems, in both the private and government sectors [1]. The problems and crises range from budget and schedule overruns to software projects that are terminated after millions of dollars are spent. The Standish Group[1] has published more staggering numbers. In two reports entitled 'Chaos' and 'Unfinished Voyages', respectively, the Standish Group estimates that in 1995 some $81 billion was spent on cancelled software projects and another $100 billion in 1996. The group's research on budget overruns indicate that more than 50% of projects will cost 189% of their original estimates. Nearly 50% of the IT executives, according to the 'Chaos' article [2], feel that there are more failures today than there were five years ago.

While the good software products and practices should be praised (as Bob Glass always does, see, e.g. his latest comments in *IEEE Software* [3]), the failures, problems and crises must also be acknowledged, and efforts must be put into improving the practice and minimize the crises. A considerable number of principles for improving the software practice have been explored in recent years, and some have proven effective in practical projects. These include software development methodologies and environments, structured and object-oriented programming, CASE tools, 4th generation languages, application generators, etc. Nevertheless, software problems and crises are not completely resolved, and a large number of organizations still suffer enormously from software budget and schedule overruns.

Many have suggested that one way of improving the practice to is emphasize proper education, and to properly educate the next generation of software engineering professionals. Some have suggested that the quality of the software engineering workplace is a direct function of the quality of software engineering education [4]. Through proper software engineering education, an education that includes the core knowledge and skills of computer science but with a primary focus on the definition, development and maintenance of software, we can succeed in our efforts to improve the quality of software produced by the next generation of software engineers, and to substantially reduce the development and maintenance costs and crises.

2. Improving the education

The debate about the most effective approach to educating the next generation of software engineers and the introduction of various software engineering education programs continues unabated. Some argue for a software engineering (SE) component or track under existing computer science (or computer engineering) programs. Others promote specialized and independent SE degrees and programs. Such SE programs focus on software development and maintenance, and cover topics that are normally not covered (in enough detail) in typical computer science programs. Examples of such topics include: project management, process improvement, testing techniques, multi-semester term projects, people and process skills, etc. Normally, computer science courses, e.g. data structures and operating systems, serve as foundation courses. Regardless of whether the SE education is provided under a computer science umbrella or as a specialized software engineering education program, the objective remains the same: to prepare a new software engineer who is exposed to the fundamentals from which technologies arise. By being exposed to the software development fundamentals, the new software engineer will be better prepared to apply the fundamental knowledge in a proper context to real problems regardless of what the popular (or 'hot') technology or product of the day is.

[1] The Standish Group International is a market research and advisory firm which specializes in mission critical software and electronic commerce. The group's web address is: www.standishgroup.com.

PII S0950-5849(98)00038-X

Software engineering education (SEE) has become the subject of many discussion and debates in recent years. The existence of many workshops, conferences and special issues of journals devoted to SEE in recent years bears witness to the importance of this relatively new discipline.[2] To address some of the challenges of SEE, the Fourth International Workshop on Software Engineering Education (IWSEE4) was held in Boston, Massachusetts (USA) on 23 May 1997. (The workshop was held in conjunction with the 1997 International Conference on Software Engineering.) More than 25 individuals from at least 10 different countries were invited to and participated in the workshop.

3. Fourth international workshop on SEE

The International Workshop on Software Engineering Eduction (IWSEE4) provided an active forum to discuss SEE, to promote collaboration between members of the academia and industry, to simulate new approaches, and to facilitate interactive exchange among the participants.[3] In addition to the presentations and debates, four Working Groups were formed during the workshop. Each of the Working Groups tackled a problem which was raised during the debates. One of the Working Groups tackled issues related to the industrial perspective. One key point of this group's report, the 'three-way' internship, was quite original and interesting. It suggested that not only is it good to send students to industry for internships, but it is equally important to send faculty to industry on internship and to invite industrial practitioners to join academia on an internship basis for team-teaching certain software engineering courses, and to provide input on course development.

This special issue of *Information and Software Technology* is devoted to some selected presentation at the workshop. Even though it is difficult to determine the current trend, teaching methods and directions in SEE curriculum across a representative sample of international academia, we did our best in selecting as many internationally diverse views as possible. The articles included here offer views and practices from educators in the US, New Zealand, Finland, Germany, UK and The Netherlands. (IWSEE4 participants from Canada, Australia and Ireland were also invited to contribute to this special issue, but were unable to meet the deadline.) A brief summary of each article follows.

Don Bagert (US) views SEE as a major sub-unit of computer science with an important role and so he examines how it can be effectively integrated into the computer science curriculum. A model based on a four-course sequence is presented and discussed, and suggestions made on how that model can be integrated into computer science programs at other institutions.

Robert Biddle and Ewan Tempero (New Zealand) discuss teaching software development by teaching the principles of reuseability. Their approach, which is based on a conceptual model of reuseability, helps simplifying program development.

Jan J. van Amstel (The Netherlands) discusses group education and training at Phillips. The Education and Training program is one of the groups within Phillips Research, and provides education and training for the professional software developers. The author describes the curriculum, organization of the software-related courses, experiences and effectiveness of these courses as SEE models for software practitioners.

Pearl Brereton (UK) and her colleagues describe distributed group working in software engineering education that was undertaken by three UK universities to provide students with the opportunity to experience group working across multiple sites using low-cost tools. Educational values of distributed cooperative working are presented at the end.

Michael Godfrey (US) shares observations and 'unexpected results' in teaching a course in software engineering to a mixed audience of undergraduate and professional Master's degree students at Cornell University.

Ann E. Kelley Sobel (US) was invited to contribute to this special issue of *Information and Software Technology* and she discusses a strategy for integrating formal methods application into several software engineering courses. This article was not presented at the workshop, but because it describes a novel approach related to SEE, it is included in this special issue.

Jochen Ludewig and Ralf Reißing (Germany) discuss a new software engineering curriculum that was launched in 1996 at the University of Stuttgart. The authors emphasize that the curriculum is new for all German speaking universities. Some background about German universities is provided, and the reason for the new curriculum as well as its goals and limitations are discussed.

Kari Alho (Finland) discusses the uses of the WWW technology in teaching, managing and administering software engineering courses at the Helsinki University of Technology.

I hope you enjoy the special issue! Let me take this opportunity to encourage you to prepare an article for the 1999 Conference on Software Engineering Education and Training (sponsored by the IEEE-Computer Society, in cooperation with the ACM, and supported by the Software Engineering Institute). The conference's purpose is to influence directions in software engineering education and training, to simulate new instructional approaches, to promote collaborations and bridges with the industry, and to generate exchanges among software engineering stakeholders. The conference will be co-located with the ACM SIGCSE Symposium on Computer Science Education. The best articles of CSEE & T'98 will be published in an special issue of the *Journal of Systems and Software* (also published by Elsevier Science).

[2] Examples of special journal issues (in addition to the current issue of *Information and Software Technology*) include *IEEE Software* (November/December 1997), *Annals of Software Engineering* (scheduled for 1999) and a special issue of *Journal of Systems and Software* planned for the best papers of the 1999 Conference on Software Engineering Education and Training (CSEE & T).

[3] The proceedings of the workshop, which included position statements, short and long articles, is available on line (in PostScript) at http://csalpha.unomaha.edu/hossein/prof.html.

References

[1] W. Gibbs, Software's chronic crisis, Scientific American 271 (3) (1994) 86–95.
[2] The Standish Group, 'Chaos', http://www.standishgroup.com/chaos.html, 1995.
[3] B. Glass, In praise of practice, IEEE Software 15 (1) (1998) 30–31.
[4] K. Beckman, N. Coulter, S. Khajehouri, N. Mead, Collaborations: Closing the industry–academia gap, IEEE Software 14 (6) (1997) 49–57.