Software Engineering Education: Prospects and Concerns of Integrating Technology

Pankaj Kamthan
Concordia University, Montreal, Quebec, Canada

INTRODUCTION

The discipline of software engineering was born out of the need of introducing order and predictability in large-scale software development. Over the last few decades, software engineering has been playing an increasingly prominent role in computer science and engineering curricula of Universities around the world (Rezaei, 2005; Tomayko, 1998).

In this article, we explore the prospects and concerns of integrating information technologies (IT) in software engineering education (SEE), both inside and outside the classroom. By IT we will mean the technologies for various activities related to information (such as acquisition, creation, communication, dissemination, processing, archival, retrieval, transformation, and so on), within the context of the Internet and the Web, unless specified otherwise.

The rest of the article is organized as follows. We first provide the background necessary for later discussion. This is followed by the prospects and concerns of systematically integrating IT in SEE and examples of use of IT in SEE, both inside and outside the classroom. Next, challenges and directions for future research are outlined. Finally, concluding remarks are given.

BACKGROUND

The discipline of software engineering (Abran, Moore, Bourque, & Dupuis, 2001) advocates a systematic approach to the sustainable development of large-scale software that aims for high quality within the given organizational constraints. To do that, having a team with diverse knowledge and skills, following a process with intermittent phases for software development, having means for quality assurance and evaluation, using models and/or documentation for communicating the progress of development across team members, making realizable (practical) choices so as to finish the product within given time and budget, and so on, are some of the traits of modern software engineering.

Like other disciplines, SEE needs to be sensitive to the variations and evolution of the social and technical environment around it. In particular, changes in IT environment need to be reflected, and to that regard, there have been calls for a reform of SEE (Frailey, 1998).

There have been some previous instances where the use of IT has been found to be useful in SEE. For example, the use of Java applets in illustrating the *dynamics* of complex algorithms has been emphasized (Kamthan, 1999), the benefits of hypertext for relating and navigating through software artifacts have been shown (Bompani, Ciancarini, & Vitali, 2002), and the use of extensible markup language (XML) for marking-up software process documents has been reported (Mundle, 2001). However, these works are limited by one or more of the following issues: the focus has been on the specifics of respective technologies rather than on the learner or on the learning process; the approach to IT integration does not appear to be systematic; and the trade-offs are seldom discussed, if at all.

A PERSPECTIVE ON INTEGRATING INFORMATION TECHNOLOGY IN SOFTWARE ENGINEERING EDUCATION

Our approach for integrating IT in SEE is based on a methodology consisting of a nonlinear and non-mutually exclusive sequence of steps as shown in Table 1.

We briefly note the following characteristics of the elements in Table 1. Firstly, we contend that the steps are necessary, but make no claim of their sufficiency. Indeed, the steps are stated at a high level and could be granularized further if necessary. Secondly, the steps 1-3 are in a bidirectional cycle (step 1 depends and is depended upon by step 2, and so on) that we exit only

Table 1. A feasibility-sensitive methodology for integrating information technologies in software engineering education

1. Deciding the Scope of Software Engineering Knowledge, Potential Information Technologies, and Educational Activities 2. Adopting a Learning Theory and a Teaching Strategy 3. Identifying and Understanding the Participants 4. Selecting and Applying Suitable Information Technologies to a Software Engineering Education Context 5. Evaluating the Effectiveness of Integrating Information Technologies in Software Engineering Education	Feasibility

when each step is adequately satisfied with respect to the others *and* is feasible. Thirdly, step 4 depends on step 1.

A detailed discussion of each of the steps is not carried out due to considerations of space. However, from a practical standpoint, we briefly highlight the significance of considering the feasibility of each of the steps 1-5. A variety of feasibility-related concerns can arise. For example, an educational activity may make sense theoretically but may be, practically, unrealizable; the adoption of objectivist or constructivist learning theory or a combination thereof (Cronjé, 2006) may seem appealing and may even be the "best" pedagogical choice, but may not be within the scope of given constraints of time and class size; it may be useful to elicit as much background on a student as possible, but privacy concerns may prevent one from doing so in its entirety; a specific IT may be an attractive option, but the software available for processing it may be proprietary and not within the given budget; and so on. The feasibility study could be a part of the overall course management activity.

The methodology is based on the *assumption* that IT can be useful in SEE. However, to give some credence to that, we must weigh the possibilities and obstacles in doing so, which we discuss next.

Prospects of Integrating Information Technology in Software Engineering Education

IT can play a role in SEE as means for teaching concepts, as means for learning concepts, or as means for performing tasks. This potential is further elaborated in the following.

Necessary alternative in a classroom. IT can give teachers alternative ways to discuss, in the classroom, the software engineering concepts that, by *nature,* are dynamic or nonlinear, and are difficult to present using traditional means. This is particularly the case with concepts related to complex structures (such as three-dimensional graphics) and evolving spatial/temporal behavior (such as iteration or recursion) in a software system.

Interactive classroom experiments. IT can be a useful tool to foster an interactive environment in a classroom. For example, a teacher could give a demonstration of a software system with a predetermined, fixed data set, and ask questions based on the variations of the data set (that will lead to unpredictable behavior of the system). In such a case, both correct and incorrect answers can contribute to the learning process.

New horizons. IT can open horizons for teachers and students to new activities and to ask/answer questions not (readily) feasible or even possible before. Using inexpensive and fast computers, it is now possible to carry out complex calculations and process very large data sets. This, for example, allows one to experiment and present the results involving software measurement in a short amount of time befitting lectures and laboratory-based tests.

New means of communication and collaboration. At times, students can find office hours insufficient or inconvenient. On the other hand, teachers may wish to keep in touch with students when they are away. For example, a teacher may need to travel out of town on a conference during the spring break, but would still like to be available for any questions from students on a software deliverable, due at the end of the break. IT can give alternative ways to teachers for communicating with their students outside the classroom synchronously or asynchronously and via client-pull/server-push. The proliferation of mobile phones with support for electronic mail and technologies for syndication has led to