Chapter X
A Methodology for Integrating Information Technology in Software Engineering Education

Pankaj Kamthan
Department of Computer Science and Software Engineering, Concordia University, Canada

ABSTRACT

The discipline of software engineering has been gaining increasing significance in computer science and engineering education. In this chapter, the goal is to describe a systematic approach toward integrating information technologies in software engineering education (SEE), both inside and outside the classroom. A methodology for integrating IT is proposed and explored in the context of SEE, particularly related to the Internet and the Web; in this context, SEE supports a heterogeneous combination of objectivism and constructivism, and aims to be feasibility sensitive. In doing so, the prospects and concerns of incorporating IT in SEE are presented. The potential of integrating IT in SEE is illustrated by examples.

INTRODUCTION

There are various sectors of society where software plays an indispensable role. This calls for special attention in the way software is developed, maintained, and used. The discipline of software engineering (Ghezzi, Jazayeri, & Mandrioli, 2003) was born out of the need to introduce order and predictability in large-scale software development. It advocates a systematic approach to the sustainable development of large-scale software that aims for high quality within the given organisational constraints.

In the last few decades, software engineering has been playing an increasingly prominent role in computer science and engineering curricula of universities around the world (Tomayko, 1998). Indeed, there has been a rise in programmes offering undergraduate and graduate degrees in software engineering (Rezaei, 2005; Surakka, 2007), including the establishment of such programmes at the author's institution.

As software engineering matures, the question of how its body of knowledge is communicated, transferred, and understood arises. Like other disciplines, software engineering education (SEE) needs to be sensitive to the variations and evolution of the social and technical environment around it. The changes in the technological environment, specifically that of information technologies, need to be reflected in education if it leads to viable opportunities and proven savings. There have been calls for a reform of SEE with a plea to give a prominent place to technology in general and IT in particular (Frailey, 1998). However, there has been little effort in the past toward precisely and objectively articulating the integration of IT in SEE, and it is this that provides the motivation for this current work.

The readership of this chapter is aimed primarily toward educators in software engineering and information systems engineering. In particular, it is therefore assumed that the reader has basic knowledge of phases, workflows, and activities in a typical software process.

The remainder of the chapter is organised as follows. First, the background necessary for later discussion is provided and the position that is taken is stated. This is followed by the introduction of a methodology for systematically integrating IT in SEE, labeled as IT4SE2. One of the steps of IT4SE2 includes the prospects and concerns of integrating IT both inside and outside the classroom. Next, some practical examples are presented, and then concluding remarks are given. Finally, challenges and directions for future research are outlined.

BACKGROUND

In this section, previous work on integrating IT in SEE is discussed. When referring to IT in this chapter, this means 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.

Impact of Information Technologies on Software Engineering Education

There have been some previous instances where the use of IT has been found to be useful in areas related to SEE. The use of Internet forums for communicating with the client during requirements elicitation and for active learning has been suggested (Parsons & Fostert, 2000). The use of the extensible markup language (XML) for marking up software process documents (Mundle, 2001) and source code (Deveaux & La Traon, 2001) has been reported. The benefits of hypertext for relating and navigating through software artefacts have been shown (Bompani, Ciancarini, & Vitali, 2002). The use of Java applets in illustrating the dynamics of complex algorithms in a classroom has been emphasised (Kamthan, 1999). The unified modeling language (UML) has emerged as a standard language for modeling the structure and behaviour of object-oriented software systems, and its use in SEE is on the rise. 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.

To (learn how to) develop software requires certain knowledge and skills. In past surveys (Lethbridge, 1998, 2000a), it was concluded that IT played a minor but relevant role in the software engineering curriculum. The prerequisite, generic, and specific skills required of a software engineer have been pointed out previously within the context of a specific research project (Seffah & Grogono, 2002). Although these skills include certain technologies, they do not include the use of the Internet or the Web in their potential, and these skills have not been placed into any known strategies of teaching or theories of learning. A