Chapter XVII
Facilitating Student–Driven Learning of Computer Programming with Technology

Alessio Gaspar
University of South Florida Polytechnic, USA

Sarah Langevin
University of South Florida Polytechnic, USA

Naomi Boyer
University of South Florida Polytechnic, USA

ABSTRACT

This chapter discusses a case study of the application of technology to facilitate undergraduate students’ learning of computer programming in an Information Technology department. The authors review the evolution of the didactic of introductory programming courses along with the learning barriers traditionally encountered by novice programmers. The growing interest of the computing education research community in a transition from instructivist to constructivist strategies is then illustrated by several recent approaches. The authors discuss how these have been enabled through the use of appropriate technologies in introductory and intermediate programming courses, delivered both online and face to face. They conclude by discussing how the integration of technology, and the switch to online environments, has the potential to enable authentic student-driven programming pedagogies as well as facilitate formal computing education research or action research in this field.
INTRODUCTION

The question of whether directed teaching methodologies are more efficient than self-directed or constructivist ones is still being debated in both global and discipline-based educational communities (Kirschner, Sweller & Clark, 2006; Sweller, Kirschner & Clark, 2007). As far as computing education research is concerned, applying constructivism in programming courses is a recurrent theme as illustrated by general studies (Kolling, Quig, Patterson, Rosenberg, 2003; Wulf, 2005). More specific reference will be considered as we review specific constructivist contributions in the remainder of this chapter. In this context, our work has been focused on studying the impact of constructivism in a specific context (i.e. university-level programming courses), rather than in “education at large”, while also further developing our understanding of how technology can actually support it. The introduction provides the context of the case study presented in this chapter (e.g. courses taught, characteristics of student population) as well as an overview of the main learning barriers encountered by novice programmers. The background section discusses how these learning barriers can be addressed by a transition from instructivist programming pedagogies to constructivist ones. We then describe and discuss the impact of specific pedagogies which have been designed for and evaluated on Information Technology (IT) undergraduate students. The next section then introduces the various technologies which were used to support our constructivist approach in both face to face and online offerings. We share the lessons learned so far and underline how the transition from face to face to online delivery resonated with these technologies. The future trend section discusses how the above-mentioned technologies might end up re-shaping the nature of computing education research. We then conclude with a summary of opinions and findings.

Context of this Case Study

Our case study focuses on the application of technology to improve the teaching of computer programming courses in the Information Technology department (IT) at the University of South Florida Polytechnic (USFP). This department offers small size classes (often in the evening) for mostly non-traditional undergraduate students (older age groups, already in the workforce). The pedagogical interventions described hereafter were deployed in two programming courses. COP2510 Programming Concepts is designed as a first programming course introducing fundamental concepts to students. During the first 5 weeks, the Raptor visual flowchart interpreter (Carlisle, Wilson, Humphries, Hadfield, 2005) is used to introduce programming building blocks and concepts while developing design skills without burdening students with the syntactical complexities of a mainstream programming language. The following five weeks are devoted to reviewing the concepts already covered with Raptor and transferring them to the Java programming language. The last five weeks explore more advanced Java concepts. This pedagogy of contents has been independently explored by others (Adams, 2007) and is based on a two-steps, syntax-light then syntax-heavy, redundant, introduction to programming. Unlike other studies, we use a fundamental-first approach (Liang, 2006), which turned out to be more welcoming to our students than the original objects-first approach we evaluated for this course (Kolling et al., 2003). Enrollment was respectively 13 and 35 students for the face to face (fall 2007) and the online offerings (spring 2008).

COP3515 IT Program Design is an intermediate programming course which requires COP2510 as pre-requisite. It strengthens students’ programming skills by (1) further developing their design and troubleshooting skills and (2) teaching a new programming language which will lead to a better