Chapter 12

Design a Computer Programming Learning Environment for Massive Open Online Courses

Ricardo Queirós
Polytechnic Institute of Porto, Portugal & CRACS/INESC TEC, Portugal

ABSTRACT

Teaching and learning computer programming is as challenging as it is difficult. Assessing the work of students and providing individualised feedback is time-consuming and error prone for teachers and frequently involves a time delay. The existent tools prove to be insufficient in domains where there is a greater need to practice. At the same time, Massive Open Online Courses (MOOC) are appearing, revealing a new way of learning. However, this paradigm raises serious questions regarding the monitoring of student progress and its timely feedback. This chapter provides a conceptual design model for a computer programming learning environment. It uses the portal interface design model, gathering information from a network of services such as repositories, program evaluators, and learning management systems, a central piece in the MOOC realm. This model is not limited to the domain of computer programming and can be adapted to any area that requires evaluation with immediate feedback.

INTRODUCTION

The evolution of e-learning in the last decades has been astonishing. In fact, e-learning seems to be constantly reinventing itself, finding new uses for technology, creating new tools, discovering new concepts. Platforms for supporting e-learning have been evolving for some years, exploring many approaches and producing a great variety of solutions.

These solutions make the learning and teaching more efficient and productive, but they usually lack effective real-time monitoring to learning process (Henda, 2013).

In the meantime many universities and institutions are using platforms for Massive Online Open Courses (MOOCs), characterised with a great diversity of topics and a huge number of enrolments. However, the real-time feedback is important for the effectiveness of MOOCs. We

DOI: 10.4018/978-1-4666-7304-5.ch012

state that novice students in an e-learning system might feel being isolated from the teachers and other students, because of the lack of essential interactions components in the system design (Jonas & Burns, 2013). This issue leads to a negative impact on the students' outcome. With well-designed synchronous virtual classrooms and collaborative tools it is possible to reduce this negative impact (Nedeva & Dineva, 2010).

This issue augments when we talk about complex domains. Learning complex skills is hard. A good example is the computer programming domain. Introductory programming courses are generally regarded as difficult and often have high failure and dropout rates (Ala-Mutka, 2005), (O'Kelly & Gibson, 2006) and (Robins et al, 2003). Many educators claim that "learning through practice" is by far the best way to learn computer programming and to engage novice students (Jonas & Burns, 2013), (Eckerdal, 2009). Practice in this area boils down to solving programming exercises. Nevertheless, solving exercises is only effective if students receive an assessment on their work. Assessing the work of students and providing individualised feedback to all students is time-consuming for teachers and frequently involves a time delay. The existent tools and specifications prove to be insufficient in complex evaluation domains where there is a greater need to practice (Rongas & Kaarna, 2004).

This paper presents a conceptual design model for learning environments regarding complex domains. Specifically, we focus on the computer programming domain. This environment uses the portal interface design model gathering information from a network of services such as repositories and program evaluators. These services will improve the responsiveness of the environment, a crucial success factor in massive courses.

The design model includes also the integration with learning management systems, a central piece in the MOOC realm, endowing this way the model with characteristics such as scalability, collaboration and interoperability.

The remainder of this paper is organised as follows: the next section presents a brief survey on integration specifications such as, the digital repositories interoperability specification and the learning tools interoperability specification. Next, we present the conceptual model of a learning environment for a complex domain such as the computer programming domain. In the following section we propose a graphical user interface for such model focusing on the user profiles and actions, screen layout and implementation details. Finally, we conclude with a summary of the main contribution of this work and a perspective of future work.

INTEGRATION SPECIFICATIONS

The current generation of e-learning platforms values the interchange of learning objects and learners' information through the adoption of standards that brought content sharing and interoperability to eLearning. Learning Objects (LO) are units of instructional content that can be used, and most of all reused, on web based eLearning systems. Despite its success in the promotion of the standardization of eLearning content, it is not enough to ensure interoperability, which is a major user concern with the existing systems. The definition of common protocols and interfaces for the communication among systems is also an important issue to address.

In the last few years there have been initiatives (Leal & Queirós, 2010) to adapt Service Oriented Architectures (SOA) to e-learning. These initiatives, commonly named e-learning frameworks, had the same goal: to provide flexible learning environments for learners worldwide. Usually they are characterized by providing a set of open interfaces to numerous reusable services organized in genres or layers and combined in service usage models.

While eLearning frameworks are general approaches for e-learning system integration, several