ABSTRACT

Teaching and learning computer programming is as challenging as difficult. Assessing the work of students and providing individualised feedback to all is time-consuming and error prone 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. At the same time Massive Open Online Courses (MOOC) are appearing revealing a new way of learning, more dynamic and more accessible. However this new paradigm raises serious questions regarding the monitoring of student progress and its timely feedback. This paper provides a conceptual design model for a computer programming learning environment. This environment uses the portal interface design model gathering information from a network of services such as repositories and program evaluators. The design model includes also the integration with learning management systems, a central piece in the MOOC realm, endowing the model with characteristics such as scalability, collaboration and interoperability. This model is not limited to the domain of computer programming and can be adapted to any complex area that requires systematic evaluation with immediate feedback.

Categories and Subject Descriptors
D.3.3 [Programming Languages]: General
K.3.1 [Computer Uses in Education]: Computer-assisted instruction (CAI)—Distance learning

General Terms
Design, Experimentation, Standardization, Languages

Keywords
Teaching Assistant, Automatic Evaluation, Programming Exercises, Interoperability, Learning Objects, MOOC

1. 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 [\textsuperscript{[1]}].

In the mean time many universities and institutions are using platforms for Massive Open Online 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 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 [\textsuperscript{[1]}]. 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 [\textsuperscript{[1]}].

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 [\textsuperscript{[1]}, \textsuperscript{[2]}, \textsuperscript{[3]}]. Many educators claim that “learning through practice” is by far the best way to learn computer programming and to engage novice students[\textsuperscript{[4]}, \textsuperscript{[5]}]. 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 [\textsuperscript{[1]}].

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 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.

2. CONCEPTUAL MODEL

Typically, a conceptual model represents entities and relationships between them regarding a specific domain. Therefore, we present the conceptual model for the design of a computer programming teaching/learning environment. The aim of this conceptual model is to express the meaning of domain concepts and the correct relationships between different concepts. The model for the the computer programming learning environment (CP-LE) is depicted by the UML component diagram in Figure 1 composed by the following concepts:

- Learning Objects Repository (LOR) to store/retrieve exercises;
- Assessment System (AS) to evaluate students exercises;
- Learning Management System (LMS) to present the exercises to students;
- Converter System (CS) to convert between different exercise formats;

![Figure 1: Conceptual model.](image)

The CP-LE has a two-fold goal: to coordinate the systems and services of this network and to interface with users, both teachers and students.

On the LMS side the choice fell on Moodle since it is a popular and open source LMS, arguably the most popular LMS nowadays [1, 2]. This LMS has made efforts to support interoperability with other e-learning systems at two levels: content (e.g. IMS CP, SCORM, IMS CC) and communication (e.g. IMS LTI). Also successfully tests were made with Sakai LMS on this network evidencing the interoperable characteristics of the proposed approach.

The LOR system selected was CrimsonHex [3] - a software for the creation of repositories of programming exercises. The exercises are described as learning objects and complying with the IMS CC specification. The repository also adheres to the IMS DRI specification to communicate with other systems. Other software for repositories were analyzed (e.g. Flori, HarvestRoad Hive, IntraLibrary) but none of them met the domain requirements for the content and communication interoperability and most of them follow a commercial development model.

The AS system selected was Mooshak [4]. Mooshak is an open source system for managing programming contests on the Web including automatic judging of submitted programs. One of the most important reasons for its selection was the support of web services.

The CS system selected was BabeLO [5]. This system converts formats of programming exercises among systems. At the time of writing this dissertation no other system was found with these characteristics.

The integration of the CP-LE component with the other systems must rely on content and communication standards. Using content and communication standards we can abstract the use of specific systems for each type of system. For instance, we can use on this network any repository as long it supports the IMS CC specification to formalize the description of programming exercises and it implements the IMS DRI specification for communication with other services.

3. A GUI PROPOSAL

In this section we propose a possible GUI for the learning environment. In the design of the Web component one of our major concerns was usability, and to promote it we followed established user interface design principles [6]. The main feature of the resulting design is the use of a single screen common to all user profiles. This type of design breaks with the traditional structure of web interfaces used by other systems [7]. To design this user interface we started with the identification of task and usage profiles, task objects and task actions. Then we selected a suitable interaction style and finally we created a screen layout.

3.1 User Profiles and Actions

At the beginning of the design process we identified the following task profiles:

- Administrator - a person responsible for the management of the system configurations such as user accounts and repository settings;
- Teacher - a person responsible for a set of activities related with the resource management such as the authoring of two type of resources: expository (e.g. video, PDF or HTML files) and evaluation resources (programming exercises) and the submission of the resources in the repository. The submission will be enforced to comply with controlled vocabularies defined in metadata standards (IEEE LOM) and possible extensions. This class of users will also receive the exercises solved by students and the automatic feedback generated by the assessment system;
- Student - a person that browses the resources and solves exercises.

We assume that users will have different usage profiles. On one hand, many will be novice or first-time users, especially among students. On the other hand, we expect some users, especially teachers and old students, to use it frequently, tending to become experts in its use. After the identification of users and usage profiles we proceeded to identify the tasks they need to perform on this interface. We clearly identified expository and evaluation resources as our task objects, each with a number of associated task actions, depending on user profiles. Task actions over resources include: viewing, downloading, solving, voting, sharing and commenting.

A typical pedagogical learning process is the classroom assignment in a Computer Science course. For instance, when a student starts solving an exercise, the CP-LE component automatically creates a project. A project contains source code and related files for building a program in a specific programming language. Thus, a set of predefined files need to be generated for the project creation. These files are related with the chosen programming language. After the automatic creation of the project the student reads the exercise description and solves it in a specialized Web editor (e.g., AceEditor). The student should test the code locally by executing the teachers test cases and is encouraged to create new ones. If new test cases are created, a validation step is performed to verify that they meet the specification defined by the teacher in the authoring phase. After testing, the student should submit the solution to the Assessment System where the submission is checked against the complete test set provided by the teacher. The report on the evaluation returned by the AS is presented to the student. The student may submit repeatedly, integrating the feedback received from the AS. In the end of this cycle, the CP-LE component reports the exercise usage data back to the repository and the grade results back to the LMS.

### 3.2 Screen Layout

To define the screen layout we sought an interaction style balancing intuitiveness and expressiveness. We first considered direct manipulation of task objects. Although it provides a convenient way to select objects, it is not possible to map all the identified task actions to basic mouse interaction (click, point and drag). We found form filling adequate for entering data for the complex tasks, such as search, collaborating with other users, playing business games or creating new materials. If new forms are created, a validation step is performed to verify that they meet the specification defined by the teacher in the authoring phase. After testing, the student should submit the solution to the Assessment System where the submission is checked against the complete test set provided by the teacher. The report on the evaluation returned by the AS is presented to the student. The student may submit repeatedly, integrating the feedback received from the AS. In the end of this cycle, the CP-LE component reports the exercise usage data back to the repository and the grade results back to the LMS.

### 3.3 Implementation

The learning environment was developed using an Ajax framework to enable the implementation of the single screen design resulting from the last section. We selected the Google Web Toolkit (GWT), an open source Java software development framework that allows for a rapid development of Ajax applications in Java. When the application is deployed, the GWT cross-compiler translates Java classes of the GUI to JavaScript files and guarantees cross-browser portability. The framework supports also asynchronous remote procedure calls. This way, tasks that require high computational resources (e.g., complex searching within the repository) can be triggered asynchronously, increasing the user interface responsiveness. The complex controls required by the selection and action areas are provided by SmartGWT, a GWT API for SmartClient, a Rich Internet Application (RIA) system. The Web component is organised in two main packages: the back-end (server) and the front-end (client). The back-end includes all the service implementations triggered by the user interface. These implementations rely on the gateway class for managing the communication with the Web services. A single class implementing the Gateway design pattern concentrates the interaction with the core component.

### 4. Conclusions

This paper presents a conceptual model for the teaching and learning process in complex domains such as computer programming. The design model is suitable for integration in MOOC platforms where there are a large number of enrolments and, at the same time, a large number of dropouts due to lack of teacher support. The adaptation to MOOC platforms is guaranteed with the integration of systems that provide automatic assessment giving to the student a higher autonomy to proceed in the course without the need to wait for teachers’ feedback.

The model could be adapted to other complex domains. Playing business games in management courses, or simulating a human patient in life sciences courses, or simulating an electronic circuit in electronics courses are examples of complex learning domains that require the use of special evaluators. Currently we have Petcha (a CP-LE component) running at SESEI - an Engineering School - with promising results.

Regarding future work we expected to include other services in this network with the inclusion of a plagiarism tool to
avoid plagiarism and ensure good scholarly practices and a resources sequencing tool. Sequencing of exercises is another topic that can be explored in the future and it is closely related with pedagogical issues during the construction of a learning scenario. Several standards appeared in recent years trying to cope this topic but fail due its complexity for e-Learning systems to implement. One research path is to deliver exercises to students dynamically according with their profiles, knowledge evolution and course goals. An intended addition is a sequencing and adaptation tool to guide the student through a collection of expository and evaluation resources. The CP-LE component will report the exercise assessment to this new tool that will use it to propose the appropriate content or exercise to the student;

5. REFERENCES