Design of a Repository of Programming Problems

José Paulo Leal *) **and Ricardo Queirós** **)

*) CRACS & DCC-FCUP / Portugal
zp@dcc.fc.up.pt

**) CRACS & DI-ESEIG / Portugal
ricardoqueiros@eseig.ipp.pt

This paper describes the design of a repository of programming problems integrated in the EduJudge project. The goal of this project is to open the repository of programming problems of the UVA Online Judge (online-judge.uva.es) to pedagogical uses in secondary and higher education. The EduJudge projects foresees an integration with existing Learning Management Systems (LMS), namely the Moodle and Quest systems. Nevertheless, our goal is to position this repository as a service provider for a larger set of systems with an automatic evaluation engine, such as a LMS or even programming contest management systems.

It is the authors perception that most of the Learning Objects (LO) repositories currently available are actually specialized search engines, and not adequate for feeding an automatic evaluation engine. Although some of these repositories list a large number of pointers to LO, they have few instances in any category, such as programming problems. Moreover, they do not provide the actual LO, only pointers, and sometimes these pointers are dangling. Last but not least, the LO listed in these repositories must be manually imported into a LMS. An evaluation engine cannot query the repository and automatically import the LO it needs.

We modeled our repository using a book library as metaphor: Learning Objects (LO) may be seen as books that the learner (the reader) accesses trough the Learning Management System (LMS). As in a book library, the repository has a catalog that provides efficient search. After selecting a LO the learner will receive an actual copy of it, as supplied by its author, not just a pointer to another service. As LO are a kind of e-books the repository will have an unlimited number of copies to lent, unlike in a physical book library.

Libraries also keep records of requisitions. It is therefore very simple easy for a librarian to know which are the most popular books, those that were never requested and the books that readers take longer to read. In our view, this kind of information will be interesting for the next generation of LMS. Instead of using fixed presentation orders of LO, as they do today, they will dynamically determine presentation orders based on information about their previous use. To accommodate these features our repository will give LMS the option record information on the use of LO and will provide statistics on this data.

Our first challenge was the definition of programming problems as LO. They must contain every information needed either by programming contest management system or by LMS with automatic evaluation of programming problems. The LO must be a package including assets and meta-data describing them. In our case assets include problems descriptions, test vectors, solution programs, etc. Our LO definition provides standard meta-data - who wrote it, when, versions, languages, etc - as well as domain specific - problem classification, role of assets, etc.

Based on this view of a programming problem repository we defined a set of design principles for our system. 1) The repository must be simple and effective. This principle lead us to identify a core component with a minimal set of features efficiently implemented. Complementary features are delegated to helper applications that connect to the core component. 2) The repository must be reusable in other contexts, with generic features available through users or applications interfaces. 3) The repository must comply existing standards. The standards include the SCORM (Sharable Content Object Reference Model) specification (data IMS CP and meta-data IEEE LOM) and interoperability with other systems based in Web Services (W3C). We modeled the repository based on these principles and we include the present also the main diagrams in this paper.