ISBN: 978-972-8924-58-4 © 2008 IADIS

MILE - MULTIFUNCTIONAL INTEGRATED LEARNING ENVIRONMENT

Boban Vesin
Bussines School, Vladimira Perića-Valtera 4, Novi Sad, Serbia

Mirjana Ivanović, Zoran Budimac, Ivan Pribela
Department for Mathematics and Informatics, Faculty of Science, Trg D. Obradovića 4, Novi Sad, Serbia

ABSTRACT

MILE is an e-learning tool to support teaching, learning and student's assessment during basic programming courses. This project integrates three existing educational systems: Mag, Svetovid and Testovid, developed by authors of the paper. Mag is the tutor for learning programming languages. Svetovid is system that helps instructors to leverage the effort of practical programming exercises and exams. Testovid is automated testing system, designed for assessing students' assignments during practical exercises. This paper describes the structure and functionalities of MILE.

KEYWORDS

e-learning, educational systems, Integrated Learning Environment

1. INTRODUCTION

It is becoming more and more evident that the requirements of commercial learning environments are too diverse to be provided by a single monolithic system (Manjon, 2002). First, we need a system related to the content production, and then these contents have to be put together into courses and delivered to learners. Then, system must provide support for communication and collaboration between students and teachers. Finally, student modelling must be implemented into the system, which will carry rich and detailed information concerning student's progress through the course, and which needs to be linked to the system itself. Therefore, these environments are more likely to be produced as an integration of a number of specialized systems. Nowadays, there could be noticed a great importance being accorded to the open-source Web technologies regarding the online assessment methods in a flexible environment. The current development in the e-learning domain is conducted to a large amount of tools which obtain easier maintenance of online tests (Buraga, 2006).

The main problem with online tutoring and testing systems is that they allow students to consult each other and share their solutions with other students from the group or even take tests instead of others (Pribela, 2007). Another problem with assessments systems is that they are not programming language independent. One of the rare systems with multi language support is TRY system, but it is a slightly outdated (Reek, 1989). The main motivation for this project is to produce environment for successful and easy learning and testing of basic principles of programming languages. This system allows students to take on-line courses, to test their knowledge, to submit their answers and receive automated feedback, to take on-line exams and be automatically assessed. These educational activities in MILE student can take in a computer laboratory or from a distant computer. Several security mechanisms integrated in MILE system allow easier monitoring of educational activities. The paper presents the technical and pedagogical goals of MILE, its principles of design, architecture and functionalities.

The paper is organized as follows. Section 2 gives overview of MILE components and section 3 concerns the overall architecture of the developed Web system. Section 4 refers to testing functionalities. Last section is dedicated to further directions of research and conclusions of the paper.

2. REVIEW OF COMPONENTS

The basic intention was to create learning environment for different programming languages that will allow use of components already developed: **Mag** - tutoring system for distance learning of programming languages; **Svetovid** - submission system and **Testovid** - automated testing system.

2.1 Mag

Mag is a tutor designed to be used by students in their first programming course (Vesin, 2007). It is an interactive system that allows students to use tutorial courses and to test their knowledge. The Mag is multifunctional educational software which fulfils three primary goals, identified by earlier exploration in this field (Jones, 2006). The first goal is to provide intelligent tutoring system for students in a platform independent manner. The second goal is to provide the teachers with useful reports identifying the strengths and weaknesses of student's learning process. Finally, the third goal is to provide a rapid development tool for creating basic elements of tutoring system: tutorials and tests. Two separated user interfaces are provided for both student (learner) and instructor (learner's mentor). All data about student and his progress in the course, as well as data about tutorials, tests and examples are stored in the system's server.

Mag supports learning by practicing and learning by samples. It combines traditional programming experience with distance education. It provides three types of learning activity: tutoring, quiz-and-feedback, and on-line programming, to meet the needs of course. Mag is used for Java programming course.

2.2 Svetovid

Svetovid is cross-platform software that helps instructors to leverage the effort of practical exercises and exams (Pribela, 2005). The motivation for developing this software was caused by problems associated with practical exercises and exams. Before the introduction of practical assessment in programming courses, students of informatics from the Department were assessed based on their solutions written on paper. There were two major problems at the time. Students were forced to write programs without access to a computer and a possibility to check them. Instructors faced the problem of evaluation of those programs by perusing the listings; looking for obvious errors and trying to understand usually very specific and clumsy solutions. Nevertheless, instructors try to give honest marks based on student's own solution. To avoid behaving as policemen, decision was to try to solve the problem as automatically as possible and prevent students from cheating. As a result a special cross-platform submission environment for secure student program submission has been developed (Pribela, 2005). he goals for Svetovid were as follows:

1. Allow students to comfortably develop and test their programs before submission.
2. Keep a log of student efforts.
3. Be flexible enough and usable for different courses including wide range of programming languages and project stage: coding, typing, program documentation...
4. Disallow students to share programs and solutions, intentionally or unintentionally.
5. Help instructors to mark student solution (i.e. program code).

2.3 Testovid

Testovid is independent educational tool with a number of useful functionalities. This testing system was implemented using Apache Ant. It allows students to test their assignments in a controlled manner. The system allows instructor to run the same tests on a set of student assignments. The results of the tests are recorded in a log file and are available to both the students and the instructor. The system accepts any file types as assignments, and the instructor has great flexibility in specifying how and what is to be tested. The system is independent of underlying platform and programming language, but as a part of MILE its primary purpose is to be used for testing student's programming solutions. The system has been used as well in assignment process of different West Balkan Universities (DAAD project, 2000-2007). The testing logic and test data are prepared by the instructors, in advance, saved in appropriate files ready for students' usage. Students can use prepared tests in order to check their solutions. When they are satisfied with results, they submit their solutions to be assessed by instructor. There are two main advantages of system usage: It is platform and language independent and Allows great flexibility in what and how will be tested, as well as the file type that can be submitted.

3. STRUCTURE AND ORGANIZATION OF MILE

In this section, organization of MILE components will be described. The primary motivation for integration of mentioned components was to build multi-functional educational system that will allow students to: Go through unlimited number of adequate tutorials; Test their understanding of new material; Do on-line programming; Submit their answers and receive automated feedback; Have final exams via Internet or in Computer laboratory. Important issues were security and easy maintenance of the system, providing instructors with direct access to student data and functionalities for altering courses and exams. Testovid and Svetovid were originally designed and developed as systems independent of programming languages. Therefore, Mag component has to be extended and it was necessary to update database of tutorials and tests with other programming languages. Mag was already provided with functionalities for adding new course materials which made this task to be solved easily. System must also provide:

- Support for teacher-student and student-student communication through chat and e-mail.
- Tools to simplify the maintenance of the system, such as the maintenance of the students' portfolio.
- Students web pages, through which learners will be able to navigate easily through the available courses.

Several problems have to be resolved in order to design final architecture of MILE:

- Mag was designed as a Java programming tutor (Svetovid and Testovid were multilingual systems),
- Testovid was primarily developed for computer lab environment,
- There was a huge difference between structures of student modelling.

3.1 Architecture of MILE

MILE modular architecture has been chosen in order to gain flexibility and to obtain extensibility. This would fit with the intended component-based architecture. It also allows the use of any tool already developed that provide services required by MILE, rather than to develop complete new one implement for every service. To assure platform independence, a Java-based implementation is adopted.

The MILE was designed having extensibility in mind (Fig. 1). All of the components described below provide interfaces for their interaction and can thus be easily overridden by a developer. All course materials students can access via Internet. Only the final exams can be executed both on-line and from computer laboratory. Functionalities included in instructor's software are presented in the right side of figure 1.

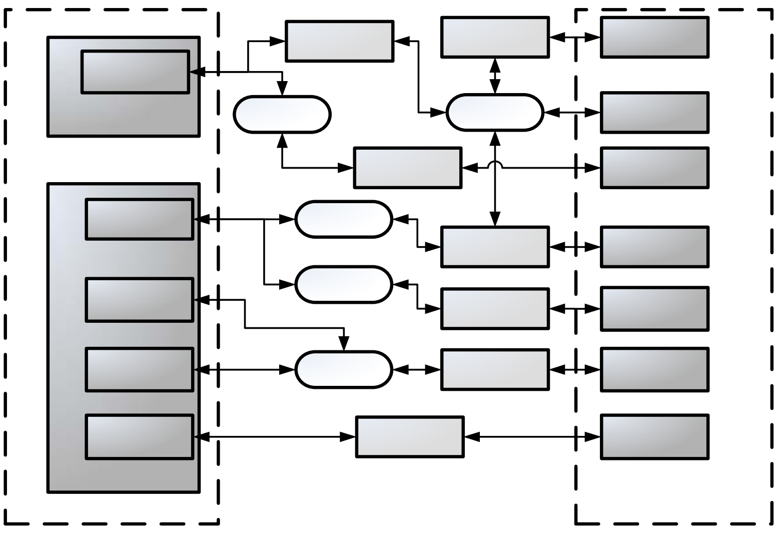

Figure 1. Architecture of MILE

The *Apache Ant Script Creator* was created as an application component for rapid development of Configuration files. Based on data from student model, system executes personalization of contents to suit individual learners. MILE tries to adapt course content to the individual learner, taking into account different levels of granularity in the information and different student knowledge levels. The primary responsibility of the *Test administration* and *Repository administration* components is providing a user interface for modifying tutorials and tests. The goal of the *Reports* tool is to collect information relate to particular student or group of students and then to present that information to instructor. System security has to prevent access to parts of the system they are not allowed to be used. Every user who wishes to use this system is required to have a username and password for login to system. MILE also includes tools for communication between students and tutors in form of chat and e-mail. The structure is general enough for teachers to design their own tutorials and other course material for different programming languages.

3.2 Testing Abilities of MILE

The testing of student's knowledge and assessments of student's assignments were given a wider attention during development of MILE. Idea was to provide student with two levels of testing:

1. Tests connected to every lesson that are used to test student's understanding of that particular lesson,
2. Final exam for overall assessment of student knowledge.

The most of currently existing e-learning and assessment systems focus on simple assessment strategies, e.g. only on single or multiple-choice questions (MCQ) with several answers, and radio-buttons to select the correct answer (Belcadhi, 2004). Beside those common assessment strategies, MILE also uses code competition questions for online programming and assessments possibilities. The system accepts any file types as assignments, and the instructor has great flexibility in specifying what is to be tested. For example when the instructor wants to automatically check if a student solution compiles successfully he should write one configuration file. File tells how many points student can award for successful compilation, how to name the aspect being tested and to give some friendly description displayed to the student. The system tries to compile all files in current directory and if all went well set the advice given to the student to an empty string. If an error occurs during compilation, advice is set accordingly. When a student invokes the system to test his/her solution, he/she will receive a message about success or failure of testing. This message contains information about the course and assignment, list of all tested aspects, and states the amount of points awarded. If the test was unsuccessful, an advice given by the Ant target will be displayed to the student. As mentioned earlier the contents presented to the student are filtered according to his/her prior knowledge and also based on his/her progress. System adopts tests to every particular student based on previously stored data in his/her student model. System makes a log file of all mistakes and successful solutions that student made and adopt next tests accordingly. For example, every test has multiple questions that are grouped by their purpose. There are questions that test understanding of code, familiarity with syntax rules, etc. Based on the student model, system discovers type of questions that caused problems to student in previous sessions and increase their number during next testing.

The implemented examine system allows students to test their assignments in a controlled manner with the instructor's test data. The instructor creates an Apache Ant script to build and run the student's project. New scripts can be created using parts from previously created scripts, thus simplifying the instructors work. When a student wishes to test his/her assignment he/she runs the system which then copies the student's files into a temporary directory, and the instructor's scripts are executed to build the project and test it. The result of each test run is recorded in a log file for the student and the instructor.

The main advantage of assessment component of MILE is that it is built on Apache Ant and thus it is modern and not dependent on programming language or specific building and compilation logic. Furthermore, the system can be used in a wide variety of situations and environments, is very extensible, modular, and can quickly adapt to new trends. The main motivation for developing MILE system and its most common usage, however, is to check student programs for compilation errors, code style guidelines adherence, implementation correctness and performance. At our Faculty, students have several courses which focus on programming exercises (using: Java, C#, Delphi, Modula-2, Scheme) as a main technique for continual assessment of practical work. During the practical exercises students work in computer laboratories on the given assignments and instructor assesses their practical skills. As this way of continual assessment is very time consuming and wearying for both, student and instructor, students can utilize MILE to check their solutions before submitting, and instructors to increase reliability and speed up the assessment. Besides on-line testing provided for students, instructor is offered a batch mode for testing of multiple submitted projects. The system keeps track of results for all students and generates a single file with all the results. Also one detailed report for each student is generated.

Proposed testing system, built on the Apache Ant, can capture details of failed build and show appropriate advice to the student in order to improve his/her solution.

4. CONCLUSION

MILE is designed to combine distance education system with submission environment with a goal to provide great degree of automated testing. By our experiences, MILE showed itself as a modern learning tool that posses a lot of features for fulfilling nowadays needs and style of learning. It provides optimal performance with use of the most appropriate architecture. Several improved tools for testing student's knowledge and tools for automating assessment of student programs are also included. Features of the system, which enables online programming, are its main advantages. All of the actions are run on server. That enables student to take courses and test his/her progress in learning from optional computer without necessity to install any specific kind of software. Various types of questions and tasks make process of student assessment easier and adequate. According to that, appropriate student model adjusts teaching and testing procedure to every particular student. MILE has been used in last semester as a learning tool for first year students at the Department as a complement to classroom teaching. It gave us opportunity to evaluate the main features of the system, the results obtained by students and the educational objectives in order to improve its functionalities and characteristics.

Students' satisfaction was, more or less, the same as in completely manual manner of assessment, which is by our opinion excellent outcome of the system. On the other hand, instructors significantly shorten time necessary for assessment of great number of students'. Based on the success of similar intelligent tutoring systems, it is also hypothesized that students will be able to learn programming skills and gain knowledge more quickly and effectively than students in traditional educational settings.

REFERENCES

Apache Ant, http://ant.apache.org/

Belcadhi L.C., Henze N., Braham R., 2004. An Assessment Framework for eLearning in the Semantic Web. *Distributed System Institut publication*, Hannover, Germany p. 6.

Buraga S., Brut M., Onacă D., 2006. An XML-based Java Application for the Management of Online Questionnaires. *I*Teach project,* Romania, p. 4.

DAAD project, Software Engineering: Computer Science Education and Research Cooperation. 2000 - 2007, http://www.informatik.hu-berlin.de/swt/intkoop/daa

Jones N., Macasek M., Walonoski J., Rasmussen K., Heffernan N., 2006. Common Tutor Object Platform – an e-Learning Software Development Strategy. *Proceedings of the 15th international conference on World Wide Web*, Edinburgh, Scotland pp. 307-316.

Manjon B.F., Sancho P. 2002. Creating cost-effective adaptive educational hypermedia based on markup technologies and e-learning standards. *Interactive Educational Multimedia,* number 4 pp. 1-11.

Pribela I., Ibrajter N., Ivanovic M., 2005. Svetovid Special Submission Environment for Students Assessment. *Proc. of Second Balkan Conference in Informatics,* Ohrid, FYROM, pp. 13-19.

Pribela I., Ivanović M., Budimac Z., 2007. Testing Almost Any Aspect Of Students' Assignments. *3rd Balkan Conference in Informatics,* Sofia, Bulgaria, pp. 173-182.

Reek, K. 1989. the TRY system -or- how to avoid testing student programs. *Proceedings of the twentieth SIGCSE technical symposium on Computer science education,* ACM Press New York, USA, pp. 112-116.

Vesin B., Ivanović M., Budimac Z., 2007. Tutoring System for Distance Learning of Java Programming Language. 10th Symposium on Programming Languages and Software Tools SPLST, Dobogókő, Hungary, pp. 310-320.