Novi Sad J. Math.
Vol. 38, No. 2, 2008, 153-170

MULTIFUNCTIONAL ENVIRONMENT FOR E-LEARNING PURPOSES[1]

Mirjana Ivanović[2], Ivan Pribela[2], Boban Vesin[3], Zoran Budimac[2]

Abstract. MILE is an e-learning tool that supports teaching, learning and student assessment within programming courses. It integrates three educational systems developed at the Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad: Mag, Svetovid and Testovid. Mag is the tutoring system for learning programming languages. Svetovid is a system that helps instructors to leverage the effort of practical programming exercises and exams. Testovid is an automated testing system, designed for assessing students' solutions during practical programming exercises. In this paper the structure and functionalities of MILE are described and discussed.

AMS Mathematics Subject Classification (2000): ???????

Key words and phrases: E-learning, educational systems, Integrated Learning Environment

1. Introduction

E-learning is the fastest growing form of education in the last decade, and it has become the most popular way of learning. Usually e-learning systems use computer networks as the delivery mechanism, and allows student to take a course anywhere and anytime, so it is widely applied not only to school's courses but also for retraining employees in companies. In view of this, there is a wide range of tools and environments (learning management systems, web-based educational environments, tutoring systems. . .) which have been used in different educational processes and distance learning manner.

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 [11]. 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, the system must provide support for communication and collaboration between students and teachers. Finally, student modeling must be implemented into the system, which will carry rich and detailed information concerning student progress through the course, and which needs to be linked to the system

[1] This work was supported by project "Abstract Methods and Applications in Computer Science" (No. 144017A), of the Ministry of Science and Technological Development of the Republic of Serbia

[2]Department for Mathematics and Informatics, Faculty of Science, Trg D. Obradovića 4, 21000 Novi Sad, Serbia, e-mail: {mira, pribela, zjb}@im.ns.ac.yu

[3]Bussines School, Vladimira Perića-Valtera 4, 21000 Novi Sad, Serbia, e-mail: vesin-boban@yahoo.com

itself. Therefore, these environments are more likely to be produced as an integration of a number of specialized systems.

Nowadays, a great importance is 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 extent by tools which obtain easier maintenance of online tests [3, 14].

The main problem with online tutoring and testing systems used in laboratories and classrooms is that they allow students to consult each other and share their solutions with other students or even take tests instead of others [14]. Another problem with assessments systems is that they are not programming language independent. The main motivation to produce such an environment was to obtain successful and easy learning and testing facilities for different programming languages and courses which have been conducted at our Department. MILE (Multifunctional integrated learning environment) is a system that combines three existing extremely useful educational components. 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 the 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 presents an overview on the related work, section 3 introduces MILE components and section 4 concerns the overall architecture of the developed Web system. Section 5 refers to testing functionalities. Last section is dedicated to further directions of research and conclusions of the paper.

2. Related work

A number of learning systems have been developed over the last ten years that include various mechanisms for assessment of student progress. Most distinguished among them are: JITS, tutoring system implemented for teaching the basics of programming in Java [12, 17, 19], and Moodle, a typical example of a successful use of a learning management system [22].

Java Intelligent Tutoring System - JITS, is a tutoring system designed for learning Java programming [19]. MILE implements some principles of tutoring and testing from the JITS system because it proved to be successful and effective. Positive results of the JITS system showed that the form of its lessons used to create a course is especially effective in teaching Java programming language [17]. Centralized architecture is implemented in MILE in contrary to distributed architecture implemented in JITS in order to start all system's actions from server side of the system [12].

Moodle is a free learning management system that enables users to create powerful learning environment full of different kinds of student-to-student and student-to-teacher interaction [21]. The main advantage of MILE system, opposite to the functionalities of Moodle, is that MILE provides possibilities of online programming rather than just presenting course material to the student. Student can write program code, compile and run his programs from remote computer without necessity of installing any software.

One very important element of e-learning is the assessment of student progress, whether for the purposes of self-evaluation or grading. In computer courses, together with assessment concerns, interest is also placed on the submission of student solutions.

These topics date all the way back to the 60's and during this long time many systems were developed that usually followed modern concepts introduced by new operating systems and new programming languages. There have also been successful contemporary attempts to address the problems arising from the usage of automatic submission and assessment tools. Unfortunately, systems like [4], [6], [10] and [18] focus on Java programming language, although there were some attempts to create a system for testing that will encompass a wider spectrum of programming languages developed in Python [1]. The only, truly, programming language independent testing system is [15]; a system built for UNIX platform, also based on command scripts, but a little outdated and without network support. Also, the approaches used in [4], [6] and [8] cannot be applied in our circumstances and in our academic environment. The focus of these approaches is on the detection of cheating, while in our opinion it should be on its prevention.

The mentioned systems rely on student honesty and provide services for the detection of student misbehavior, mainly plagiarism. Our experience shows that we cannot rely on student's honesty as the main problem at some universities is the lack of sanctions. In such environment students do not have anything to lose, and thus resort to plagiarism without the fear of consequences. Furthermore, the provide services are not perfect and can only detect plagiarism if the original work is also submitted.

Svetovid successfully resolves some of the key assessment problems by providing an integrated environment that must be used for the submission. This environment represents first line of defense and makes cheating difficult if not impossible for the student.

By designing our system in a different way, we moved the focus from plagiarism detection to plagiarism prevention. We believe that this is a much better approach than using some of the modern system relying on plagiarism detection, like BOSS [10], CourseMaster / Ceilidh [7] or ASAP [24]. Both Svetovid and Testovid are platform independent (written in the Java programming language and based on Apache Ant tool [23]), network-oriented systems and are created to be compatible with many programming languages, thus adhering to modern requirements.

3. Overview of MILE components

The basic idea and intention was to create a multifunctional learning environment which integrates existing learning systems that could be used for different courses and a wide range of programming languages. Three components already implemented are chosen to make the basic structure of MILE:

1. Mag - tutoring system for distance learning of programming languages,
2. Svetovid - submission system and
3. Testovid - automated testing system.

In the following subsections these components are described in more detail.

3.1. Mag - tutoring system

Mag is a tutoring system designed to help students in learning programming languages in different courses [20]. It is an interactive system that allows students to use teaching material prepared for programming languages within courses and to test their knowledge, as well.

The Mag is multifunctional educational software which fulfils three primary goals, identified by earlier exploration in this field [9]. 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.

Mag supports learning by practicing and learning by samples. It combines traditional programming experience with distance education. The System provides a learner with a more efficient and convenient way of taking a distance programming course. It provides three types of learning activity: tutoring, quiz-and-feedback, and on-line programming, to meet the needs of programming course.

In spite of the fact that this system is designed and implemented as a general tutoring system for different programming languages, the first completely proposed and tested version was for an introductory Java programming course.

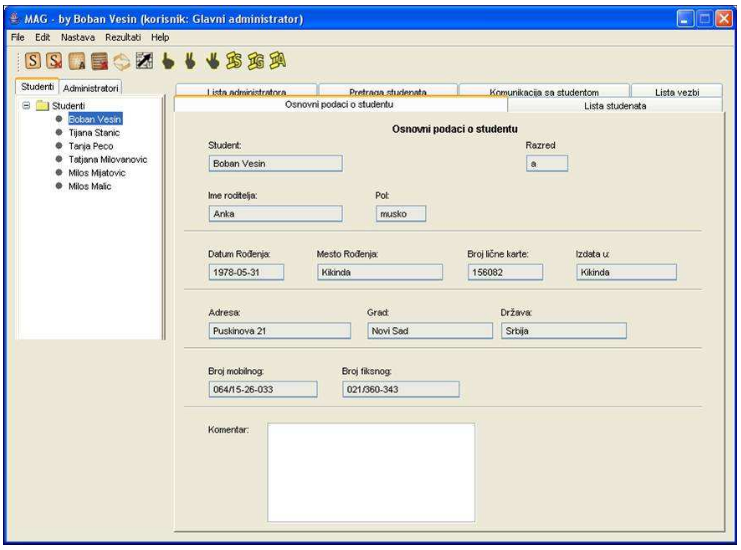
Preliminary design of the *Mag* system was based on several requirements that every on-line learning system for a programming language should have [20]:

- easy-to-access tutorials for students
- separated user interfaces for students and their mentors
- various examples for every particular lesson (learning module)
- different tests for every particular lesson that can be adjusted to particular student

- online programming, compiling and running of programs
- functionalities for adding new lessons, examples, and tests possibilities for communication between students and mentors
- functionalities for easy monitoring of student's work
- summaries and reports about student's work.

The system recognizes two main roles, intended for two types of users:

- **mentors -** their role is to be the lesson and student database administrator, to track progress of students learning and help them with their assignments and
- **students -** they are taking the Java programming course and will be using the system in order to gain certain knowledge.


MAG - by Boban Vesin (korisnik: Glavni administrator)
File Edit Nastava Rezultati Help
Studenti Administratori
Lista administratora
Pretraga studenata
Komunikacija sa studentom
Lista vezbi
Osnovni podaci o studentu
Lista studenata

Studenti
- Boban Vesin
- Tijana Stanic
- Tanja Peco
- Tatjana Milovanovic
- Milos Mijatovic
- Milos Malic

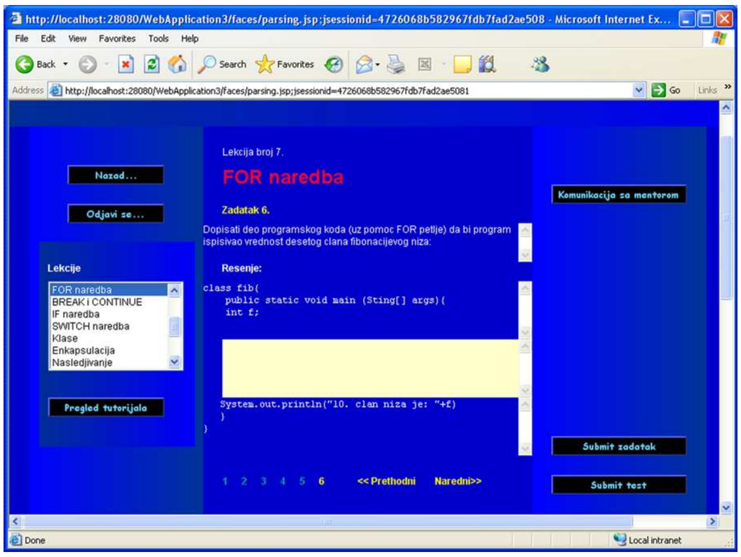
Osnovni podaci o studentu

Student: Boban Vesin
Razred: a
Ime roditelja: Anka
Pol: musko
Datum Rođenja: 1978-05-31
Mesto Rođenja: Kikinda
Broj lične karte: 156082
Izdata u: Kikinda
Adresa: Puskinova 21
Grad: Novi Sad
Država: Srbija
Broj mobilnog: 064/15-26-033
Broj fiksnog: 021/360-343
Komentar:

Figure 1: Mentor interface

Therefore, two separate user interfaces are provided for both student (learner, 2) and instructor (learner's mentor, 1). Instructor's interface helps in the process of managing data about a learner and course material. Student's interface is a series of web pages that provide two options: taking lessons and testing student's knowledge. 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.

The proposed architecture has numerous benefits: platform-independence, weightlessness and scalability. Students do not need to install software on their


http://localhost:28080/WebApplication3/faces/parsing.jsp;jsessionid=4726068b582967fdb7fad2ae508 - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:28080/WebApplication3/faces/parsing.jsp;jsessionid=4726068b582967fdb7fad2ae5081
Go Links
Nazad...
Odjavi se...
Lekcije
FOR naredba
BREAK i CONTINUE
IF naredba
SWITCH naredba
Klase
Enkapsulacija
Nasledjivanje
Pregled tutorijala
Komunikacija sa mentorom
Submit zadatak
Submit test
1 2 3 4 5 6 << Prethodni Naredni>>
Done
Local intranet

Lekcija broj 7.

FOR naredba

Zadatak 6.

Dopisati deo programskog koda (uz pomoc FOR petlje) da bi program ispisivao vrednost desetog clana fibonacijevog niza:

Resenje:

```
class fib{
    public static void main (Sting[] args){
    int f;

   System.out.println("10. clan niza je: "+f)
   }
}
```

Figure 2: Student interface

own machine and do not need a high-speed network connection to use *Mag*. Other benefits include fast execution, since all processing is done on the J2EE server that typically has much faster and more efficient hardware than typical PCs.

3.2. Svetovid - submission system

Svetovid is a cross-platform software that helps instructors to leverage the effort of practical exercises and exams [13]. It is named after the pagan Slavic god that sees everything.

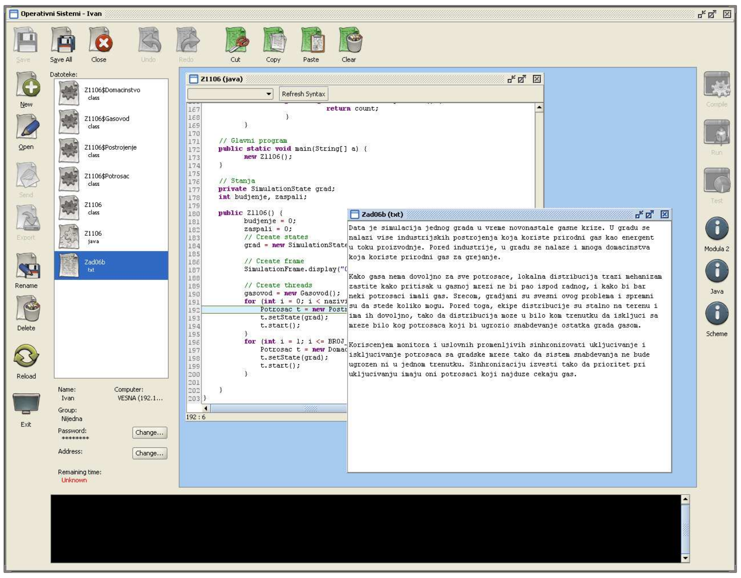
The motivation for developing this software was prompted by the problems associated with practical exercises and exams within different programming courses at our Department. Before the introduction of practical assessment in programming courses, students of informatics from the Department were assessed based on their hand-written solutions. 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, for at maximum three hours. 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. This was tedious for both students and instructors, and efficient only for small assignments.

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 [13].

The goals, to be fulfilled by the Svetovid system were as follows:

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

The Svetovid system is designed and implemented with an easy-to-use client environment for program development by incorporating most common IDE (Integrated Development Environment) ideas (3). Using some sample input and output data, prepared by the instructor for the preliminary testing, that is available in advance, students can test their programs under exactly the same conditions the instructor will. This ensures that the program will work the way the student expects it to.



```
                        return count;
                }
        }

    // Glavni program
    public static void main(String[] a) {
        new Z1106();
    }

    // Stanja
    private SimulationState grad;
    int budjenje, zaspali;

    public Z1106() {
        budjenje = 0;
        zaspali = 0;
        // Create states
        grad = new SimulationState
        
        // Create frame
        SimulationFrame.display("(

        // Create threads
        gasovod = new Gasovod();
        for (int i = 0; i < nazivi
            Potrosac t = new Postr
            t.setState(grad);
            t.start();
        }
        for (int i = 1; i <= BROJ_
            Potrosac t = new Domac
            t.setState(grad);
            t.start();
        }

    }
}
```

```
Data je simulacija jednog grada u vreme novonastale gasne krize. U gradu se
nalazi vise industrijskih postrojenja koja koriste prirodni gas kao energent
u toku proizvodnje. Pored industrije, u gradu se nalaze i mnoga domacinstva
koja koriste prirodni gas za grejanje.

Kako gasa nema dovoljno za sve potrosace, lokalna distribucija trazi mehanizam
zastite kako pritisak u gasnoj mrezi ne bi pao ispod radnog, i kako bi bar
neki potrosaci imali gas. Srecom, gradjani su svesni ovog problema i spremni
su da stede koliko mogu. Pored toga, ekipe distribucije su stalno na terenu i
ima ih dovoljno, tako da distribucija moze u bilo kom trenutku da iskljuci sa
mreze bilo kog potrosaca koji bi ugrozio snabdevanje ostatka grada gasom.

Koriscenjem monitora i uslovnih promenljivih sinhronizovati ukljucivanje i
iskljucivanje potrosaca sa gradske mreze tako da sistem snabdevanja ne bude
ugrozen ni u jednom trenutku. Sinhronizaciju izvesti tako da prioritet pri
ukljucivanju imaju oni potrosaci koji najduze cekaju gas.
```

Figure 3: Svetovid client environment

Also, students work on their solutions in computer labs and can submit their assignments whenever they are satisfied with them. If the student never explicitly submits his solution, it is automatically submitted after the assignment deadline has passed. After all student solutions are submitted the instructor can check them within the same environment under the same conditions.

The system incorporates an extensive logging mechanism. Separate log files are kept for each day, and each course. By reviewing log files the instructors can find: the exact time and the ID of a computer from which the student saved/loaded a file; the number and success of compilations and program runs, with exact time and computer used for each action. Thus, using this information, the instructor can monitor student's progress, or find potentially suspicious students (the ones who did nothing, while their submitted program compiles and works flawlessly).

By means of password protected virtual directories placed on a server, accessible to students only from the Svetovid client, disallows students to share programs and solutions, intentionally or unintentionally.

The Svetovid system is being used to automate compiling and running of Modula 2, Java, and Scheme programs for courses in Operating systems, Programming languages, Data structures and algorithms and Computer graphics.

3.3. Testovid – testing system

Testovid was originally developed as a part of Svetovid [14]. But in the meantime it grew into an independent educational tool with a number of useful functionalities. The goal of the system is to leverage the effort of assessing assignments during practical exercises especially at our Department. The system has been used as well in assignment process of different West Balkan Universities (participants of our joint DAAD project, 2000-2008) [25].

The type and number of files that the system can accept is not restricted. The system simply makes a copy of all student files and runs prepared test scripts. Also, the system has no limit on the amount of tests being conducted, and during one run multiple aspects of the assignment can be tested.

Testovid system can be used to check virtually any aspect of student solution, limited only by teachers' imagination and proficiency with Java and Apache Ant. Most notably, for programming solutions, system can test compilation, code style, solution correctness (using input-output sets, comparing with correct solution, unit testing, black box and white box testing, and so on), and check common programming pitfalls.

However, the system is not limited to checking programming solutions, it can be used to check document syntax and grammar (in student essays or seminar papers), compare documents or their parts, evaluate quiz answers, process images, and perform many other tasks supported by Apache Ant.

Furthermore, Testovid is able to incorporate any Apache Ant library and is extensible as Ant is. Custom Ant libraries with new functionalities can be implemented and used to extend current Testovid capabilities. Also any third party libraries that are already developed can be included immediately.

The testing logic and test data are prepared by the teacher, in advance, saved in appropriate files ready for students' usage. Students can use many prepared tests in order to automatically check their solutions. Most importantly, teacher uses the same tests under the same conditions, in process of marking solutions.

Testovid system is designed to run in two modes: interactive and batch mode.

Interactive mode is used when a student wishes to test his assignment. If the student thinks that his assignment is finished he can invoke the system from a workstation in computer laboratory or using web interface from home. The result from prepared test script is given to the student and also saved in a log file for later reference (Table 1).

```
Course:  Object oriented programming
Assignment:  Java beans Assignment no 2
Student:  Baranovski Nenad

Test:  Compilation
Failure:  Compilation failed, there were syntax errors.

Test:  Geters and seters
Failure:  Unable to test due to compilation errors.

-+-+-+-+-+-+-+-+-

Total:  0 points.
```

Table 1: Log file with student results

On the other hand, if the system is not intended to be used interactively by students there is a possibility for instructor to test multiple projects at once.

The batch mode can be used if students are working on their assignments at home or if interactive test ability is not desired or not allowed. In this case all assignments must be collected before start of testing. Usually, the assignments will be gathered by a separate submission system, and manually saved under Testovid directory tree. Assignments are tested one at a time and results are saved in log files. A file containing results for all students is maintained (Table 2) together with separate result reports, one for each student.

There are two main advantages of the system usage:

1. It is platform and language independent and
    (a) Allows great flexibility in what and how will be tested, as well as the file type that can be submitted to testing.

This testing system was implemented using Apache Ant, and 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 instructor's work. The results of the tests are recorded in log files and are available to both the students and the instructor. The Testovid system is independent of underlying platform and programming language, but

```
Course:  Object oriented programming
Assignment:  Java beans Assignment no 2

Student                 Score
----                    ---
Bajcetic Vladimir       2 points
Baranovski Nenad        0 points
Barjaktarovic Marko     0 points
Blagojevic Katarina     1 point
Boskovic Marko          1 point
Bratic Radana           1 point
Curcin Nada             2 points
...
Jovanovic Jelena        1 point
Knezevic Branko         2 points
Komaromi Filip          0 points
Kovacevic Miroslav      0 points
Krickovic Valentina     1 point
Krizan Jan              2 points
Kuzmanovic Milos        0 points
Lackovic Ivan           2 points
```

Table 2: Log file with results for all students

as a part of MILE, its primary purpose is to be used for testing student's programming solutions.

4. Overall architecture and organization of MILE

The primary motivation for integration of mentioned components was to build a multi-functional educational system that will allow students wide range of useful opportunities:

- 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. MILE provides instructors with the user interface for direct access to student data and options for quick and easy update of courses and exams.

Testovid and Svetovid were originally designed and developed as systems independent of programming languages. Although Mag was initially considered as a general tutoring system, its first version supported only course for programming in Java. Therefore, Mag component needed to be extended and it was necessary to update database of tutorials and tests with other programming languages in order to gain flexibility and provide courses for programming in other languages. That flexibility would allow the use of MILE as a multifunctional environment for learning, testing and student assessment for various courses and programming languages at our Department. Mag was already provided with functionalities for adding new course materials which made this task to be solved easily.

The system also provides several additional functionalities:

- 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 or students' and tutors' registration.
  - Students web pages, through which learners will be able to navigate easily through the available courses, and specific course tutorials, as well.

Several major problems had to be resolved in order to design final architecture of MILE. Mag was designed as a general architecture for learning programming languages, but the pilot version was realized as Java programming tutor (on contrary Svetovid and Testovid were multilingual systems) [20]. Therefore, it was necessary to adjust existing components and obtain smoothly functioning of full architecture. Second, Testovid was primarily developed for computer lab environment. Current version of MILE allows only testing from distant computer with multiple-choice and code completion questions. Our future goal is to allow final examining (Apache Ant tool based) outside of computer laboratory. Finally, there was a huge difference between structures of student modeling. Student model in Svetovid was presented with one data record that included just basic information about student. For MILE, we needed more complex student model that will contain data about all learning sessions, testing results and knowledge level for every student.

Database of tutorials and tests was extended to support courses for Modula 2 and Scheme programming languages; access to them from remote computers is allowed and one unique form of student modeling is adopted.

4.1. Architecture of MILE

In order to gain flexibility and obtain extensibility module architecture for MILE is chosen. This fits 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 a completely new one implement for every service. To ensure platform independence, a Java-based implementation is adopted.

The Architecture of MILE is presented in Figure 4. It was designed having extensibility in mind. 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, from a distant computer. Only the final exams can be executed both on-line and from computer laboratory. Numerous functionalities included in the instructor's software are presented in the right-hand side of Figure 4.

The *Apache Ant Script Creator* was created as an application component for rapid development of Configuration files. Based on data from student model, the system executes personalization of contents to suit individual learner's preferences. 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 that will be presented to students. The goal of the *Reports* tool is to collect information related to a particular student or group of students and then to present that information to the 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 the form of chat and e-mail. The structure is general enough for teachers to design their own tutorials and other material for different courses on programming languages.

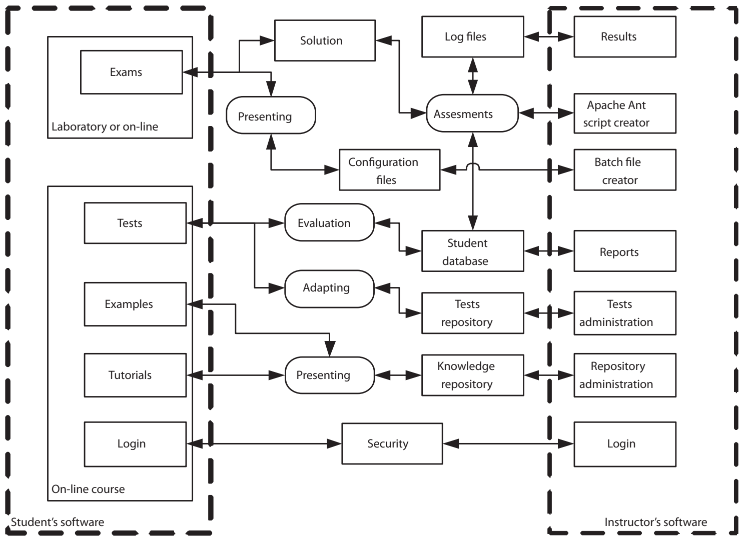


Figure 4: Architecture of MILE

Mentor's software gives the mentor an opportunity to get insight into student's work with numerous reports, to create new tutorials, examples, tests and lessons, to communicate with students, etc.

Student's software provides easy-to-use access to all functionalities of the system. It contains web pages that can be explored in any order, with tutorials, examples and options for testing, submission of answers and communication with the mentor.

Students have their choice in using the system for learning in two different modes.

In the first mode, a student is guided through the course, and order of lessons is predefined in advance. A student must pass the test for the current lesson before proceeding to the next one. In the worst case, if the student could not pass the test during several attempts, he/she can cancel the learning activity.

In the second mode, a student can have an insight into all course material without limits and test his/her knowledge without predefined order. The student can skip current lesson at anytime and can choose another lesson by his/her preferences. This mode is important if the student needs to be reminded of the past material, to get a quick preview of upcoming lessons or to take lessons in the order he/she desires.

4.2. Organization of course material

The available materials for Java course is divided into learning objects. Learning objects (LO) are small units of learning, ranging from 2 to 15 minutes, according to SCORM (Sharable Courseware Object Reference Model), the ADL standards framework [16]. A LO is constructed from Media Assets, such as paragraphs of text or html, screen titles, captions, video, animation, diagrams, and sound narration [5].

Learning objects in MILE are presented in the form of lessons. Every lesson contains three basic parts: tutorials, examples and tests. An unlimited number of examples and tests are attached to every lesson. The system provides the mentor with the possibilities of adding new lessons as well as new tutorials, examples and questions for existing tests. All these elements have a standardized structure that allows the implementation of the user interface forms for entering new lessons, tests and examples.

Java lessons of the *MILE* are divided into several subtopics: introduction, syntax, loop statements, execution control, specific types and classes. Their interaction is shown in Figure 2. Nodes present groups of lessons, and arrows present order of execution.

Every tutorial contains explanation of concepts and appropriate syntax rules for the material presented in the lesson. After the tutorial, the student is provided with several examples connected to the lesson. If a student wants to exercise more examples, he can choose *additional examples* option.

At any stage of learning, anywhere within a lesson, student can take one of prepared tests for that lesson. This can be very beneficial to the student as (s)he can check his/her progress at any time and focus on parts (s)he has missed or has not fully understood. With this approach the student can also check his/her precognition before taking a lesson or consolidate his knowledge of learned concepts after a lesson. This availability of tests can also be used to facilitate learn-by-example approach.

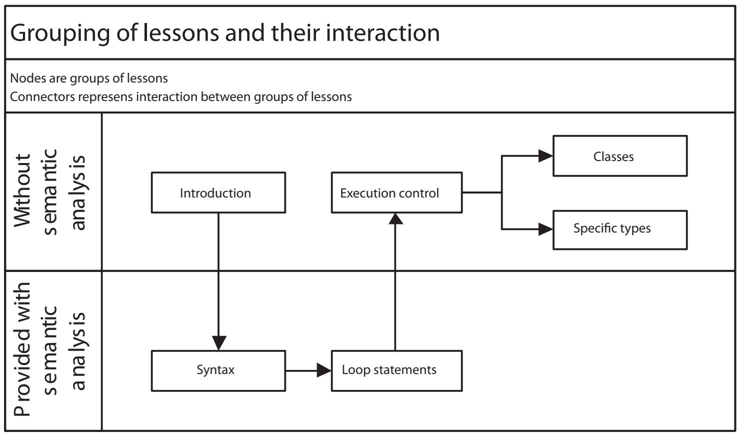

Grouping of lessons and their interaction
Nodes are groups of lessons
Connectors represens interaction between groups of lessons
Without semantic analysis
Provided with semantic analysis
Introduction
Execution control
Classes
Specific types
Syntax
Loop statements

Figure 5: Grouping of lessons and their interaction

5. Testing functionalities of MILE

Testing and assessment of students' knowledge are important parts of all modern e-learning systems. Students must have insight in their learning progress during entire course. Therefore, testing functionalities must be implemented into the system.

The testing of student knowledge and assessments of student solutions were given a wider attention during the development of MILE. The 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 the overall assessment of student's knowledge.

Tests connected to every lesson. 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 [2]. Beside those common assessment strategies, MILE also uses code competition questions for online programming and assessments possibilities that are implemented using Apache Ant. The system accepts any file types as assignments, and the instructor has great flexibility in specifying what is to be tested and how.

We may look at an example where the instructor wants to automatically check if a student solution compiles successfully. The instructor should write one configuration file telling how many points to award for successful compilation, how to name the aspect being tested and to give some friendly description displayed to the student. The list of actions to be performed during testing has also to be set up.

When a student invokes the system to test his/her solution, he/she will receive a message about success or failure of testing that particular program. 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 the progress of the learner. The system adopts tests to every particular student based on the previously stored data in his/her student model. The system makes a log file of all mistakes and successful solutions that the 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, the system discovers type of questions that caused problems to student in the previous sessions and increase their number during next testing.

Final exam for overall assessment of student's knowledge. 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 proposed multifunctional learning environment has a lot of advantages. The main advantage of assessment component of MILE is the consequence of that it is built on Apache Ant. 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 be quickly adapted to new trends. The MILE system supports testing of any aspect of student solutions written in any programming language. Even more, it is not limited to assessment of programming solutions; it can accept textual files, images, and other documents too.

The main motivation for developing the MILE system and its most common usage, however, is to check student's programs for compilation errors, code style guidelines adherence, implementation correctness and performance. At our Department, students have several courses which focus on programming exercises (using following programming languages: 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 the 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, the instructor is offered a batch mode for testing of multiple submitted students projects. The system keeps track of results for all students and generates a single file with all the results. Apart from that, a detailed report for each student is generated and students' model updated.

6. Conclusion

MILE has been designed to combine distance education system with submission environment in order to provide great degree of automated testing possibilities. By our experiences, MILE appeared to be a modern learning tool that possesses a lot of features for fulfilling nowadays needs and style of learning. It provides optimal performance with the 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. The features of the system, which enable online programming, are its main advantages. All of the actions are run on the server. This enables student to take courses and test his/her progress in learning from optional computer without the necessity to install any specific kind of software. Various types of questions and tasks make the process of student assessment easier and appropriate. According to that, appropriate student's model adjusts teaching and testing procedure to every particular student. The MILE was organized in such a manner that provides further extension of the system and smooth integration of the other components.

MILE has been used in the last semester as a learning tool for the 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 the completely manual manner of assessment, which is by our opinion an excellent outcome of the system. On the other hand, instructors significantly shorten the time needed for assessment of a great number of students' solutions, and, as a consequence, they managed their work and duties in a more efficient and more systematized way.

Students are also allowed to use the system from their home and therefore provided with the possibility to learn and practice programming without time limitations.

References

[1] Amelung, M., Piotrowski, M., Roesner, D., EduComponents: Experiences in EAssessment in Computer Science Education. ITiCSE'06, Bologna, Italy, June 26-28, 2006.

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

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

[4] Dempster, J., Web-based assessment software: Fit for purpose or squeeze to fit? Computer Assisted Assessment, Issue 6, University of Warwick, 1999.

[5] Gallenson, A., Heins, J., Heins, T., Macromedia MX: Creating Learning Objects. Macromedia Inc, 2002, p. 34.

[6] Hawkes, T., An Experiment in Computer-Assisted Assessment. Computer Assisted Assessment, Issue 6, University of Warwick, 1999.

[7] Higgins, C., Symeonidis, P., Tsintsifas, A. The marking system for coursemaster. Proceedings of the 7th annual conference on Innovation and technology in computer science education, ACM Press, 2002, pp. 46-50.

[8] Joanna, B., Supporting Computer-based Assessment. Teaching and Learning Directorate, University of Luton, 1999.

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

[10] Luck, M., Joy, M., Computer-Based Submission and Assessment in BOSS. Interactions Online Journal, Issue 6, 1999.

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

[12] Murray, T., Intelligent Tutoring Systems Architecture. Proceedings of the Third International Conference on Intelligent Tutoring Systems, Montreal, 1996, pp. 469-511.

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

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

[15] Reek, K. 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, 1989, pp. 112-116.

[16] Shepherd C., E-Learning's Greatest Hits. Above and Beyond, 2003, p. 192.

[17] Sykes, E.R., Java intelligent tutoring system model and architecture. Proceedings of American Association of Artificial Intelligence Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments, Menlo Park, CA, 2003, pp. 187-193.

[18] Sykes, E. R., Franek, F., A prototype for an intelligent tutoring system for students learning to program in Java. Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education, Rhodes, Greece, 2003, pp. 78-83.

[19] Sykes, E. R., Franek, F., An Intelligent Tutoring System Prototype for Learning to Program Java. The third IEEE International Conference on Advanced Learning Technologies (ICALT'03), Athens, Greece, 2003, pp. 485-494.

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

[21] White, G. L., A Theory of the Relationships between Cognitive Requirements of Computer Programming Languages and Programmers' Cognitive Characteristics. Journal of Information Systems Education, USA, Vol 13(1), 2002, p. 8.

[22] William, H. Rice IV., Moodle E-Learning Course Development. Packt Publishing Ltd, 2006, p. 250.

[23] Apache Ant, `http://ant.apache.org/`

[24] Automated, System for Assessment of Programming, `http://www.elframework.org/projects/asap`

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

Received by the editors October 30, 2008