Annales Univ. Sci. Budapest., Sect. Comp. 31 (2009) 75-92

LEARNING MANAGEMENT SYSTEM FOR PROGRAMMING IN JAVA

B. Vesin, M. Ivanović, Z. Budimac

(Novi Sad, Serbia)

Abstract. The increasing popularity of the World Wide Web and the Internet has affected computer assisted learning that is now turning into web-based learning. Web-based learning can take place anywhere, at any time, through any computer and without necessarily the presence of a human tutor.

Mag is a learning management system designed to be used by students in their first programming course. It provides three types of learning activities: tutoring, quiz-and-feedback and on-line programming, to meet the needs of programming course. Students are provided with numerous Java server pages (JSP) for learning and testing their gained skills.

Mag supports learning by practicing and learning by samples. It combines traditional in-class programming experience with learning through a tutoring system. The system provides learners with a more efficient and convenient way by taking an on-line Java programming approach.

This paper presents the technical and pedagogical objectives of *Mag*, its principles of design and architecture.

1. Introduction

Currently, the demand for tutoring systems is growing at an amazing rate. They gain such strong popularity and acceptance due to the need to:

- increase student performance,

- provide opportunity to deepen cognitive development, and
- reduce acquisition time for the student.

The wider objectives of learning a programming language includes the following:

- let the learner become familiar with the programming language,
- provide training for the abilities to correct syntax errors in source code and
- develop skills to solve various programming problems.

On-line learning has been the fastest growing form of education in the last decade and it has become the most popular way of learning. Learning management systems (LMS) hold a high position in learning technology within higher education. The intent of an LMS was to enable administrators and mentors to manage the learning process. LMSs use computer networks as a delivery mechanism and they allow students to take courses anywhere and anytime, so it is widely applied not only in classroom courses but also for retraining employees in companies.

Tutoring systems are, in many aspects, very similar to human tutors. Based on cognitive science and artificial intelligence, they have proven their worth in multiple ways in multiple domains in education [1]. More than ever, this is an important area for institutions with a lot of students wishing to obtain programming skills, and where it is difficult to provide personalized instruction needed [2].

The content of an on-line course usually can be represented in different ways: text, graphics, audio, animation and video [3]. The learning activities of most learning management systems include pre-test, on-line tutorials, exercises, quizzes, forums, etc. Web-based education is very popular and the environment needed for its setup is rather easy to be created by applying modern internet technologies: dynamic Web pages, personal profile and media streaming. However, a programming language course relies more on hands-on training, the programming skills are developed through those activities of learning by practicing, learning by debugging and learning by samples. Therefore, it is more difficult to adapt learning management systems for a programming language course.

In this paper the *Mag* system has been proposed. It is designed to meet the requirements of a programming course and provides three types of learning activities: automated tutoring, quiz-and-feedback and on-line programming through mentoring. A student can do hands-on practices as well as learning activities in the virtual environment.

The next section discusses the traditional learning objectives and learning strategies of a programming course. Section 3 is dedicated to the related work.

Section 4 describes the design and architecture of *Mag* system. Section 5 briefly describes knowledge representation and testing abilities of the system. Finally, conclusions are drawn and future work considered.

2. Traditional learning objectives and strategies of a programming course

The objectives of learning a programming language includes several aspects [4]:

- introducing the learner to the programming language and training to obtain abilities to correct syntax errors in source code,
- developing skills to fix bugs in a program and
- improving the logic analysis and reasoning abilities of problem solving.

In a programming course, one of the first things to learn is the syntax of the language and the semantics of its constructs. If a programmer is unfamiliar with the syntax and semantics of the programming language, his/her program will contain syntax errors and it will be inefficient. Modern programming tools can speed program development with an integrated editor, compiler, debugger, GUI (*Graphical User Interface*) tools, code formatting, etc [4]. Without a development tool, programming is tiring and debugging is more difficult.

A compiler for a particular programming language helps the programmer to detect syntax errors, but semantic errors show up in a program when it runs after it is compiled. If a programmer has deficiencies in analyzing and reasoning, s/he will fail in coping with complex problems. Even more, the programmer will be unable to find bugs when semantic errors in his programs appear.

Improving the student's reasoning abilities in problem solving is the most important but difficult objective of a programming course [5]. There are no shortcuts to achieve this goal. The programming skills can only be developed by repeatedly practicing the programming cycle of writing, compiling, debugging and testing the program. Our research goal was to bring together recent developments in the fields of on-line tutoring systems using artificial intelligence to construct an effective tutor which will help students to learn how to write programs in Java programming language.

3. Related work

A number of new tutoring systems have been developed over the last ten years, among them Pepite, learning environment for mathematics, JITS - tutoring system implemented in Moodle for learning the basics of programming in Java - which is a typical example of a successful use of a learning management system. Pépite software [6] consists of three modules: module for students (PepiTest); module for analysis (PepiDiag); module for teachers (PepiProf).

Similar modules have been developed in *Mag*, but modules for analysis and for teachers are bonded in one in *Mag*, in form of a Windows application. Its main purpose is to help a teacher in monitoring students as well as providing them with numerous feedbacks.

PepiTest is a module of the software dedicated to the students. It proposes 22 exercises derived from the paper and pencil tasks and it gathers students' answers to particular problems. It contains closed questions and multiple-choice questions or more interactive answering techniques (for instance matching clickable parts of graphics with limited numbers of possible answers). Multiple-choice questions are also one of the main mechanisms for testing the students' knowledge in *Mag*.

PepiDiag is a module which analyses closed natural language answers and algebraic expressions. *Mag* does not contain natural language answers, therefore it is possible to automatically analyze all provided answers and grade students' knowledge automatically (the help of human tutors is not necessary in process of testing). It also contains student's software in form of collection of Java Server pages (JSP) where a student takes tutorials and tests his acquired knowledge.

PepiProf establishes the student's profile and presents it to the teacher. It also provides an interface to modify student's answers in order to allow the teacher to control the software coding and to correct or complete it when necessary. Apart from that, *Mag* provides possibilities of automatic adaptation of course to every particular student. Description of adaptation will be described further in paper. Different kinds of reports about the students, groups and lessons are also offered.

Java Intelligent Tutoring System - JITS is a tutoring system designed for learning Java programming [7]. JITS allows learners to do hands-on practices as well as those learning activities supported in asynchronous tutoring systems. The learning activities are designed to accomplish the following three objectives:

- train the student to master the development tools through simulation,
- train the students to become familiar with the Java language through different types of quiz-and-feedback and
- improve programming skills (coding, compiling and testing of Java applet).

Mag implements 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 form of its lessons used to create a course is especially effective in teaching Java programming language [8]. Centralized architecture is implemented in *Mag* in contrary to distributed architecture implemented in JITS in order to start all system's actions from server side of the system [3]. The major improvement that *Mag* introduces is that it is a web-based tutoring system that guides student through the course.

Moodle is a free learning management system that enables users to create powerful, flexible and engaging online learning experiences [9]. This system provides students and teachers with a more active and engaging role in the process of studying. It contains web pages that can be explored in any order, courses with live chats among students and teachers, forums where users can rate messages on their relevance, online workshops that enable students to collaborate and evaluate each other's work, inquiries that let the teacher evaluate what students think of the progress of the course, directories set aside for students to upload and share their files, etc. All of these features create an active learning environment, full of different kinds of student-to-student and student-to-teacher interaction.

The main advantage of *Mag* system, opposite to the functionalities of Moodle is that *Mag* 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.

4. Design of the *Mag* system

Preliminary design of he *Mag* system was based on several basic system requirements that every on-line learning system for a programming language should have [10]:

- separated user interfaces for students and their mentors,
- easy-to-access tutorials for students,
- 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,
- summaries and reports about student's work,
- functionalities for easy monitoring of student's work,

- functionalities for adding new lessons, examples and tests,
- possibilities for communication between students and mentors.

The system is intended to be used by two types of users (two main roles exist):

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

4.1. System architecture

System architecture of *Mag* was designed in order to meet all of the mentioned requirements (Figure 1). It is a form of centralized architecture that was proven to be the most effective for constructing tutoring systems [3]. All actions are done on system's server and all student data are also kept there.

Two separated user interfaces are provided for both student (learner) and his/her mentor. The mentor's interface is a windows application with functionalities for managing data about students and that of course materials. The student's interface is a series of web pages that provides options for taking lessons and testing student's knowledge. All data about the students and their progress in the course as well as data about lessons are stored in the system's server.

The proposed architecture has numerous benefits. It is platform-independent, lightweight and scalable. Students do not need to install software on their 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.

The architecture uses a JDBC (Java DataBase Connectivity) connection to an external database which stores and retrieves specific information about students, including their progress history and performance statistics.

Previous research in the field of tutoring systems has identified five major components (Figure 2) as the most effective way for designing those systems: the student module, the pedagogical module, the domain knowledge module, the expert module and the communication module [3]. Proposed architecture of system *Mag* contains all of them in various forms.

Student module is used for student modelling. It stores information that is specific to each individual learner. At a minimum, such module tracks how well a student has mastered the material being taught. A possible addition to this is also to record misconceptions. Since the purpose of the student module is to provide data for the pedagogical module of the system, all of the information gathered should be available for the pedagogical module.

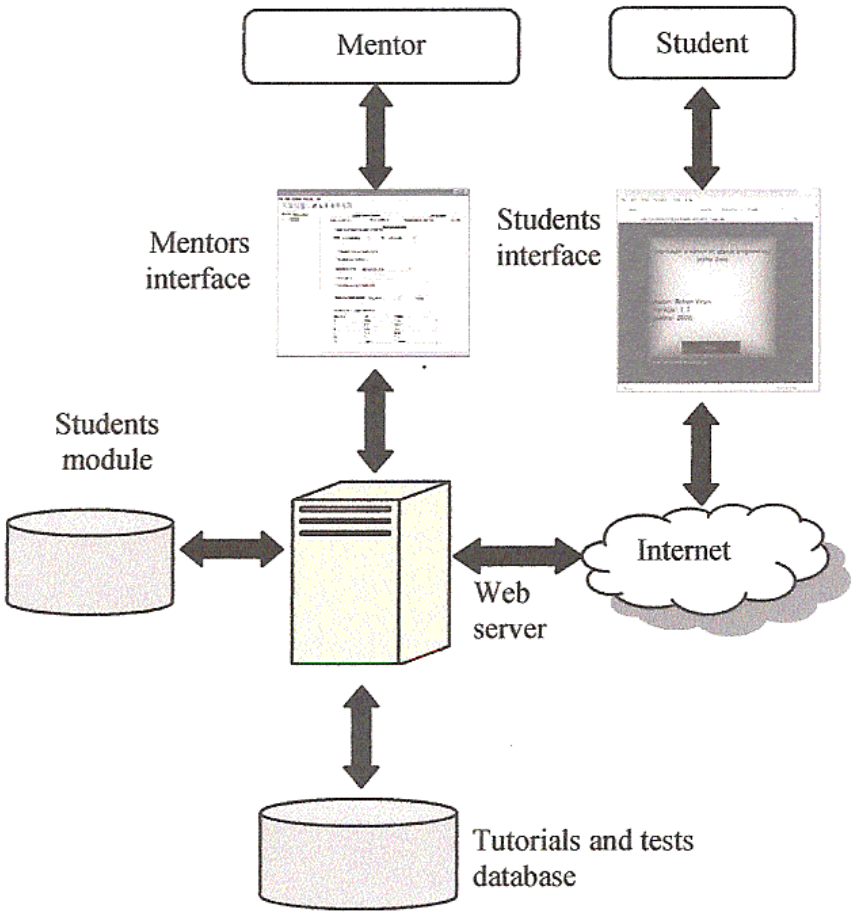


Figure 1. Scheme supported in *Mag*

The student module of a *Mag* system contains student's personal data as well as data about his progress in the course. All necessary data are stored on the server:

- student's personal data,
- number and percentage of correct and incorrect answers for every particular test that the student submitted,
- data about problems in solving every particular type of question,
- student's grades for every particular lesson.

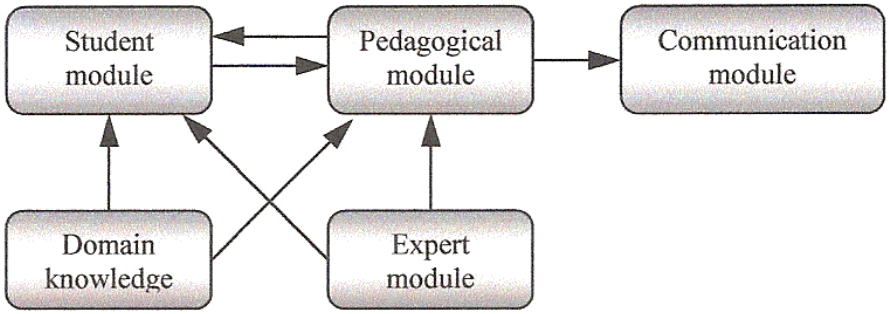


Figure 2. Interaction of components in tutoring systems

Using this data, the system can adopt tests to every particular student by choosing the percentage of several different types of questions that will be given to the student. Principles of tests adaptation will be presented later in paper.

Pedagogical module. This component provides a model of the teaching process, by controlling when to review, when to present a new topic and which topic to present. As mentioned earlier, the student module should be used as input to this component, so the system's pedagogical decisions reflect the different needs of each student. Data about student's progress and ability for solving various types of tasks are used for making decisions which lessons and what kind of tests, system will present to the student.

Domain knowledge. This component contains knowledge from the domain of teaching. Generally, it requires significant knowledge engineering to represent a domain so that other parts of the pedagogical module can access it.

Domain knowledge of *Mag* system contains a collection of 19 lessons (learning modules). Lessons are divided into several areas, based on the type of programming language elements they present. Every lesson contains three basic parts: tutorials, examples and tests. The mentor is completely responsible for lessons and their structure and for the essential elements. From time to time he/she makes revisions of lessons and can decide to add new lessons, improve elements of already existing lessons: tutorials, examples and tests, and maybe delete some of them.

Communications module. Interactions with the learner, including the dialogue and the screen layouts, are controlled by this component. Main problem that appears is how the material should be presented to the student in the most effective way.

Mag contains series of JSP pages as a layout for tutorials and testing student's

knowledge. Order of pages presentation is proposed by the communication module. On the other hand, students can change the order at any time or choose what activity will be taken next. Request for communication with mentor and detailed review of current progress is always available for the student.

Expert module. This module is similar to the domain module. However, it is more than just a representation of the data; it is an appropriate way of presenting knowledge to the student [11]. Most commonly, it is a part of the system that is capable of solving problems from the domain. By using an expert module, the tutor can compare the student's solution to the expected solution, pinpointing the subject in which the learner had difficulties.

In many programming problems, there are often many possibilities for expert's solutions. The mentor may provide one solution to the problem but there may be almost unlimited number of other solutions that are equally suitable. Expert module of *Mag* tries to evaluate student's answers rather than compare those to some given solutions. Its first task is to discover syntax errors in the student's code and afterwards to compile and run program and analyze results of execution.

4.2. User interface of *Mag*

The design of the student's user interface is a significant factor in designing computer-based tutoring systems [12] therefore, it was given careful consideration during the design of *Mag*. *Mag* system has two major parts:

- subsystem for mentors and
- subsystem for students.

Mentors' subsystem gives the mentor an opportunity to get insight into student's work with numerous reports, to adapt the course to every particular student, to create new tutorials, examples, tests and lessons, to communicate with students, etc. Web pages for students must provide easy-to-use access to all functionalities of the system. A Web page that gives a student an opportunity to test his knowledge is shown in Figure 3. List of lessons' titles are shown in the box list at the left side of interface. Every test, committed to the particular lesson contains several multiple-choice questions and problems for code completion, which are all shown at the central part of the web page. Options for communication with mentor and submission of answers are at the right side of the page.

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.

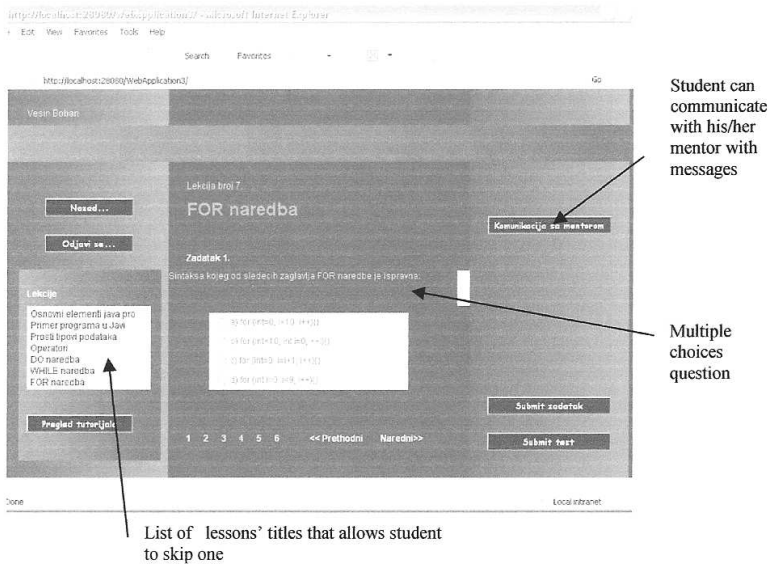


Figure 3. The *Mag*'s user interface

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 order he/she desires.

5. Organization of lessons and testing the students knowledge

Available materials for Java courses are 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 [13]. A LO is constructed from Media Assets, such as paragraphs of text or html, screen titles, captions, video, animation, diagrams and sound narration [14].

In *Mag*, learning objects are presented in form of lessons. As we mentioned

before, every lesson contains three basic parts: tutorials, examples and tests. Unlimited number of examples and tests are attached to every lesson. The system provides the mentor with possibilities of adding new lessons as well as new tutorials, examples and questions for existing tests. All these elements have standardized structure that allowed the implementation of the user interface forms for entering new lessons, tests and examples.

Lessons of the *Mag* are divided into several areas: introduction, syntax, loop statements, execution control, specific types and classes. Their interaction is shown in Figure 4. 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 could choose additional examples option. At any stage of learning and processing particular lessons student can decide to start the process of testing.

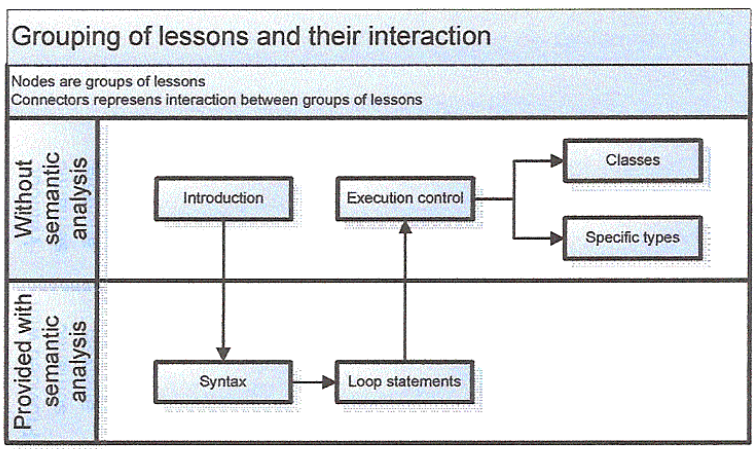


Figure 4. Grouping of lessons and their interaction

Some tutoring systems require the teacher to match problems with corresponding solutions [15]. That operation in *Mag* is automated. *Mag* is also designed to give the teachers an easy way to create the tutorials, examples and tests, and interconnect them.

Tests connected to every lesson consist of three types of questions:

- **Multiple - choice of syntax.** This type of test is used to ask the learner to trace the correct sample code.

- **Multiple - choice of execution.** This type of test is used to ask the learner to choose correct result after execution of offered code portion.

- **Code completion.** Problem is presented in the form of a skeleton program with the specific area for entering appropriate code snippet according to program specification.

The flow chart of *Mag* activities connected to testing of student's knowledge of current lesson is shown in Figure 5. The student is presented with a tutorial for next lesson after he/she provides correct answers to the current lesson.

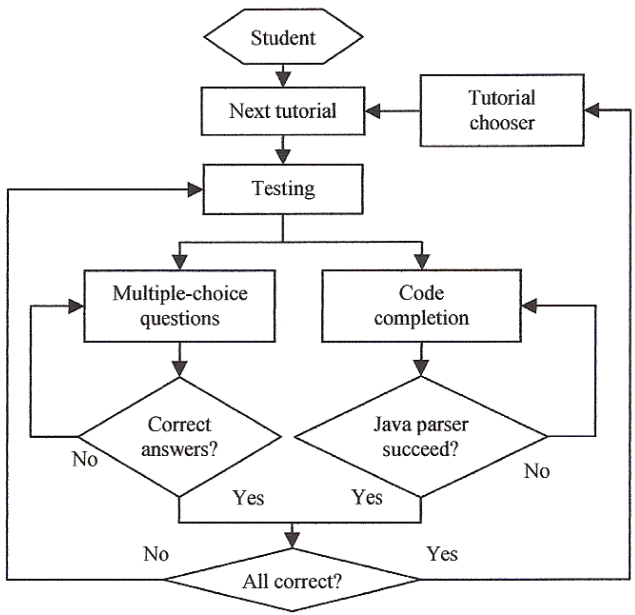
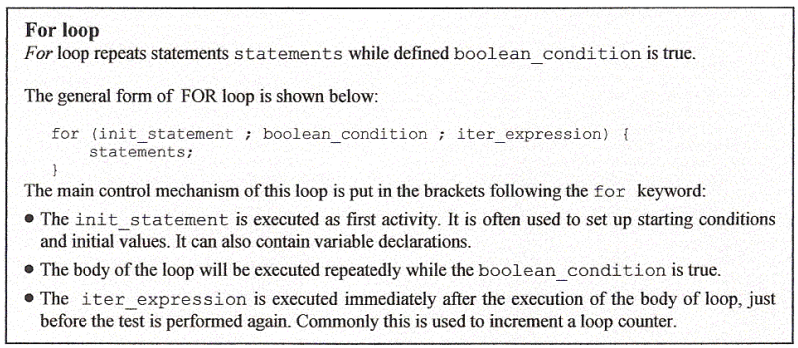


Figure 5. Flow chart of *Mag* modul for testing of student's knowledge

There is no predefined order for submitting answers to questions for any test. If a student has doubts about certain question, he can skip it and come back later to try to answer it again. He/she could quit the course in every moment and *Mag* will record and remember his current results and progress in the appropriate student module.

6. An example of a lesson

A lesson that presents basics of *FOR* statement will be used as an example. Tutorial for this lesson is presented in Figure 6.



For loop

For loop repeats statements `statements` while defined `boolean_condition` is true.

The general form of FOR loop is shown below:

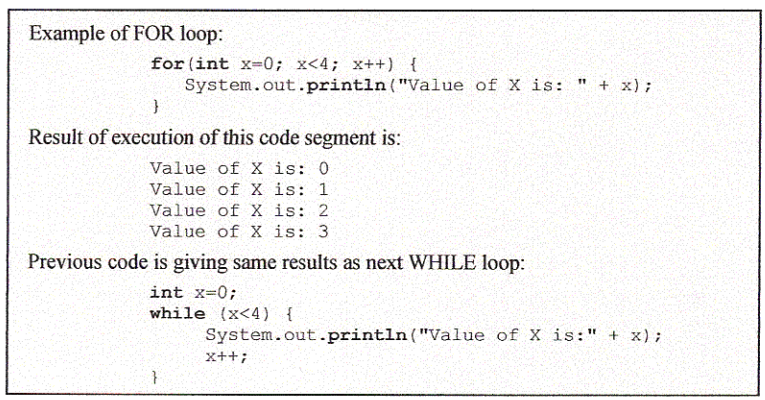
```
for (init_statement ; boolean_condition ; iter_expression) {
    statements;
}
```

The main control mechanism of this loop is put in the brackets following the `for` keyword:

- The `init_statement` is executed as first activity. It is often used to set up starting conditions and initial values. It can also contain variable declarations.
- The body of the loop will be executed repeatedly while the `boolean_condition` is true.
- The `iter_expression` is executed immediately after the execution of the body of loop, just before the test is performed again. Commonly this is used to increment a loop counter.

Figure 6. *FOR loop* lesson tutorial

There are several examples for every lesson. An example of FOR loop lesson is shown in Figure 7.



Example of FOR loop:

```
for(int x=0; x<4; x++) {
   System.out.println("Value of X is: " + x);
}
```

Result of execution of this code segment is:

```
Value of X is: 0
Value of X is: 1
Value of X is: 2
Value of X is: 3
```

Previous code is giving same results as next WHILE loop:

```
int x=0;
while (x<4) {
     System.out.println("Value of X is:" + x);
     x++;
}
```

Figure 7. Example of *For loop* lesson

Every test consists of six questions. The system forms an appropriate test by choosing questions from appropriate database. As mentioned earlier, there are

three types of questions. They are used to test student's knowledge of syntax, understanding of code sample or his/her programming skills. Distribution of every question type is depending on previous work of that particular student based on his/her data from student module. If the learner had problems with solving specific question type that means that he/she has problem with some of previously mentioned activities, therefore, percentage of that type of question in the next test is increased. That allows student to practice activities that he was poor at during previous sessions. An example of multiple-choices question for testing knowledge of syntax for this lesson is shown in Figure 8.

An example of multiple-choices of syntax question for this lesson is shown in Figure 8.

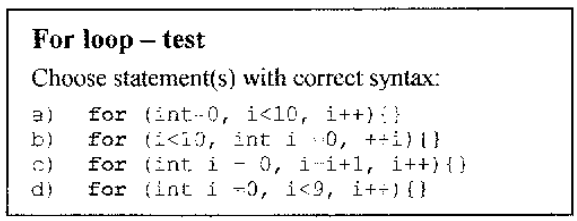


Figure 8. Multiple-choice question

Example of code completion question is shown in Figure 9.

In the code completion type of assignments, the learner tries to solve a problem and adds the missing code. The system analyzes the given solution and hints for missing statements, inappropriate syntax, wrong data types, etc. There are a number of appropriate hints for each incorrect response category, which will lead the student to the correct answer. Hints are presented to the student as long as his/her program does not match the form expected. Since the list of implemented hints is limited, and although the algorithm and tutoring process will result in a source program that will pass compilation process, there is no guarantee that it will return correct results. Implemented hints present some kind of scaffolding implemented in this system. Future plans are to increase the size and effectiveness of scaffolding used as system evolves.

In the previous example, after entering the code, the learner can try to correct the syntax errors (if there are any) according to the system hints. The learner can continue with the execution of the program if errors are not found. After the execution, the system compares the result of the execution with the result expected.

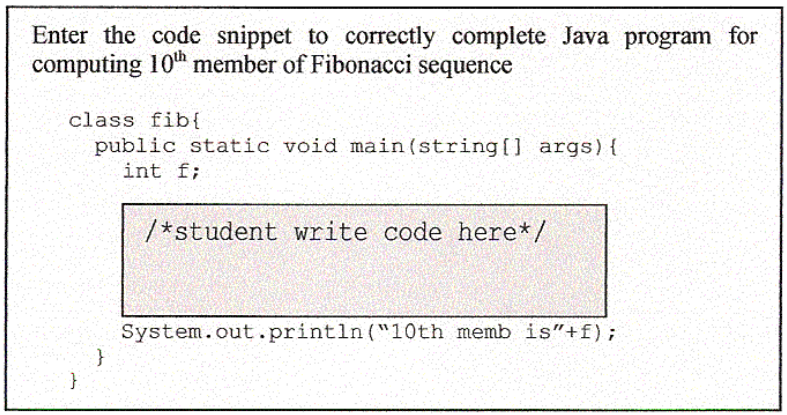


Figure 9. Skeleton program

The most important element during testing is the cooperation between system components in order to efficiently grade and update appropriate student's module. Sequence diagram presents communication between components during testing (Figure 10).

7. Conclusion

Mag is designed to implement distance education with experiences from traditional way of learning programming languages. It can provide student with a more convenient and efficient way to proceeding a Java programming course. When *Mag* is compared to the other similar tools, we can say that it possesses a lot of features in order to fulfil the present needs for the learning of programming [1]. It provides fast performance with the use of the most appropriate centralized architecture. Several improved tools for testing student's knowledge and tools for guiding students towards answers enables successful teaching process [17].

The features of the system, which enables online programming, are its main advantages. All of the actions are run on the server. That enables a student to take courses and tests from different computers without necessity to install any specific kind of software. Various types of questions and tasks in tests it make easier to adequately grade the students and to build the student module. Appropriate student module makes easier adaptation of the remaining process of

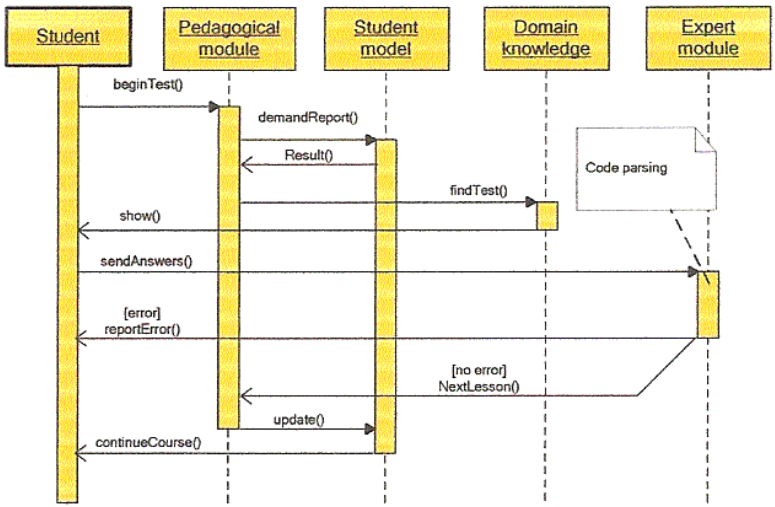


Figure 10. Communication between components during testing

teaching and testing to every particular student in a way previously mentioned.

Mag system was tested by a group of 20 students at the Department of Mathematics and Informatics, at the Faculty of Science, Novi Sad, Serbia, during the last semester. Although they used the system informally, their positive response and comments were gained. They liked the possibility of taking a course anytime and anywhere. Guiding through the online course material helps them, as they say, to speed up their progress and to finish the intended course with pleasure. In addition, they were very satisfied with the possibility to be tested after appropriate tutorials. It helps them to realize which parts of tutorial they did not understand.

The applied system architecture brought several benefits to practical use of the system. It provides possibilities for easier integration of *Mag* system into wider learning environment. The basic idea is to merge *Mag* and several existing systems and create learning environments for different programming languages. Starting point will be Svetovid and Testovid, systems that were also developed at our Department. Svetovid is a software that helps instructors to lever the effort of practical exercises and exams [16]. Testovid is an automated testing system, designed for assessing students' assignments during practical exercises [17]. The result of integration of these components will be a system that will allow the students to take on-line courses, test their knowledge, submit their programs and

receive automated feedback, take on-line exams and be automatically assessed.

This architecture has also made it possible to add new components in a simple way. Therefore, *Mag* possesses potential to become multifunctional learning management system.

The implementation and employment of *Mag* system in our institutions is a significant additional possibility for the students. It, also, has the potential to be applied to many programming courses at the university and college level. It is also quite well timed considering the tremendous growth of web-based educational tools, and the fact that Java has become an extremely popular programming language all over the world.

References

[1] **Anderson J.R., Corbett A.T., Koedinger K.R. and Pelletier R.,** Cognitive Tutors: Lessons learned, *The Journal of the Learning Sciences*, **4** (1995), 167-207.

[2] **Forbus K.D. and Feltovich P.J.,** *Smart machines in education,* Cambridge, MA, MIT Press, 2001.

[3] **Murray T.,** Intelligent tutoring systems architecture, *Proceedings of the Third International Conference on Intelligent Tutoring Systems, Montreal, 1996*, 469-511.

[4] **Blank G., Parvez S., Wei F. and Fang S.,** A Web-based ITS for OO Design, *Proc. 12th Int. Conf. on Artificial Intelligence in Education, Amsterdam, 2005.*
`www.cse.lehigh.edu/~cimel/papers/AIEDworkshop-poster.pdf`

[5] **Wu S., Tsai S. and Yang P.,** Javalab - A Java Tutorial and Programming Laboratory System, *Exploring Innovation in Education and Research*, iCEER, Taiwan, 2005, IV.4-1-IV.4-5.

[6] `http://pepite.univ-lemans.fr/`

[7] **Sykes E.R. and 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*, 485-494.

[8] **White G.L.,** A theory of the relationships between cognitive requirements of computer programming languages and programmers' cognitive characteristics, *Journal of Information Systems Education*, **13** (1) (2002), 8.

[9] **Rice W.H.**, *Moodle e-learning course development,* Packt Publishing Ltd, 2006.

[10] **Franek F. and Sykes E.R.,** Inside the Java intelligent tutoring system prototype: Parsing student code submissions with intent recognition, *Proc. IASTED Int. Conf. on Computers and Advaced Technology in Education, Innsbruck, Austria, January 2004*, 613-618.

[11] **Woolf B.,** AI in education. *Encyclopedia of Artificial Intelligence*, John Wiley & Sons, Inc., New York, 1995, 434-444.

[12] **Bednarik R., Moreno A. and Myller N.,** Program visualization for programming education - Case of Jeliot 3, *Association for Computing Machinery New Zealand Bulletin*, **2** (2) (2006).
http://is-alt.massey.ac.nz/acmnz/bulletin/vol2/issue2

[13] **Shepherd C.,** *E-Learning's greatest hits*, Above and Beyond, 2003.

[14] **Gallenson A., Heins J. and Heins T.,** *Macromedia MX: Creating learning objects*, Macromedia Inc., 2002.

[15] **Belcadhi L.C., Henze N. and Braham R.,** *An assessment framework for eLearning in the Semantic Web*, Distributed System Institute publication, Hannover, Germany, 2004, 11-16.

[16] **Pribela I., Ibrajter N. and Ivanović M.,** Svetovid special submission environment for students assessment, *Proc. of Second Balkan Conference in Informatics, Ohrid, FYROM, November 17-19, 2005,* 228-237.

[17] **Pribela I., Ivanović M. and Budimac Z.,** Testing almost any aspect of students' assignments, *3rd Balkan Conference in Informatics, BCI'2007, Sofia, Bulgaria*, 173-182.

B. Vesin
Bussines School
Vladimira Perića-Valtera 4
21000 Novi Sad, Serbia
vesinboban@yahoo.com

M. Ivanović and Z. Budimac
Department for Mathematics and Informatics
Faculty of Science
Trg D. Obradovia 4
21000 Novi Sad, Serbia
{mira, zjb}@im.ns.ac.yu