Tutoring System for Distance Learning of Java Programming Language

Boban Vesin[1], Mirjana Ivanović[2], Zoran Budimac[2]

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

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

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 human tutor.
Mag is a tutor designed to be used by students in their first programming course. It provides three types of learning activity: tutoring, quiz-and-feedback, and on-line programming, to meet the needs of programming course. Student is provided with numerous JSP web pages for learning and testing the gained skills.
Mag supports learning by practicing and learning by samples. It combines traditional programming experience with distance education. System provides learner with a more efficient and convenient way of taking a distance Java programming course.
This article presents the technical and pedagogical goals of *Mag*, its principles of design and architecture.

Key Words: Web-based education, Distance learning, Educational software, Java programming language

1 Introduction

Currently, the demand for a web-based tutoring system is growing at an amazing rate. Tutoring systems are gaining such strong popularity and acceptance due to the following reasons:

- higher student performance,
- growing opportunity to deepen cognitive development, and
- reduction of acquisition time for the student.

The performance objectives of learning a programming language include the following:

- let the learner be familiar with the programming language,

- training for the abilities to correct syntax errors in source code and
- developing the skills to solve various programming problems.

Distance learning is the fastest growing form of education in the last decade, and it has become the most popular way of learning. Distance learning systems use webs or 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 to company's training projects.

Tutoring systems are, in many respects, very similar to human tutors. Based on cognitive science and an artificial intelligence, tutoring systems have proven their worth in multiple ways in multiple domains in education [1]. More than ever, this is an important area for institutions where there are more students wishing to learn to program, and where it is difficult to provide personalized instruction that they need [2].

The content of a distance learning course can consist of media such as text, graphics, audio, animation, and video [3]. The learning activities of most distance learning systems include pre-test, on-line tutorials, exercises, quizzes, forum, etc. A distance learning program is very popular and easy to be created by applying modern internet technologies such as dynamic Web pages, personal profile and media streaming. However, a programming language course more relies 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 apply distance learning system for a programming language course.

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

The next section discusses the traditional learning objectives and learning strategies of a programming course. Section 3 presents related work and similar tutoring systems. Section 4 describes the design and architecture of *Mag* system. Section 5 briefly describes knowledge representation and testing abilities. 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 usually include next three aspects [11]:

- introducing the learner with the programming language and training for the abilities to correct syntax errors in source code,
- developing the skills to fix the bugs in a program and
- improve the logic analysis and reasoning ability of problems solving.

In programming course, one of the first things to learn is the syntax of the language and the semantics of its constructs [12]. If a programmer is unfamiliar with the syntax

and semantics of the programming language, his program will contain syntax errors and it will be inefficient. Modern programming tools can speed development with an integrated editor, compiler, debugger, visual designers, code formatting, etc. Without a development tool, programming is tiring and debugging is more difficult.

The compiler helps a programmer detect syntax errors, but semantic errors show up in a program's behaviour after it is compiled. If a programmer is deficient in analyzing and reasoning, he will fail to cope with complex problems and it will be difficult to debug when there are semantic errors in his programs.

Improving the student's reasoning ability for problems solving is the most important but difficult objective of programming course. 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, cognitive science, and artificial intelligence to construct an effective intelligent tutor to help students learn to write program 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 and JITS, tutoring system for learning basics of programming in Java.

Pépite software [4] 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 Windows application. It's main purpose is to help teacher in monitoring students as well as providing him with numerous reports.

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

PepiDiag is the module which analyses closed natural languages answers and algebraic expressions. *Mag* does not contain natural language answers, therefore it is possible to automatically analyze all of provided answers and grade students' knowledge without necessary help of human tutors. It also contains student's software in form of JSP pages collection where student takes tutorials and test his new gained 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 easy adapting course to every particular student. Different kinds of reports about student, groups, and lessons are also offered.

Java Intelligent Tutoring System - JITS is a tutoring system designed for learning Java programming [5]. JITS allow learners to do hands-on practices as well as those learning activities supported in asynchronous distance education system. 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 JITS system because it proved itself as a successful and effective. Positive results of JITS system showed that Cognitive theory used to create course is especially effective in teaching Java programming language. Major improvement that *Mag* introduces is that it is web-based tutoring system that guides student through course.

4 Design of *Mag* System

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

- separated user interfaces for students and administrators of system (student's mentors)
- easy-to-access tutorials for student
- various examples for every particular lesson
- different tests for every particular lesson that can be adjusted to particular student
- online programming, compiling and executing 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.

4.1 System Architecture

System architecture of *Mag* was designed in order to meet all of requirements [13] (Figure 1).

Two separated user interfaces are provided for both student (learner) and administrator of system (learner's mentor). Administrator's software is windows application with functionalities for managing data about students and course material. Student's software is series of web pages that provides options for taking lessons and testing student's knowledge. All of data about student and his progress in course as well as data about lessons, tests and examples are stored on system's server.

The proposed architecture has numerous benefits. It is platform-independent, lightweight and scalable. The student will never need to install software on his machine and he will not need a high-speed network connection to use *Mag*. Other benefits include fast execution, as all processing is done on the J2EE server that typically have much faster and more efficient hardware than typical PCs.

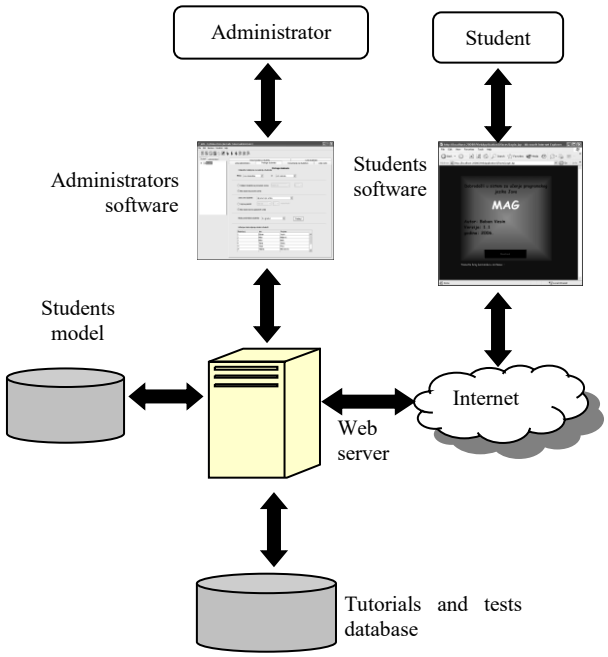

Fig. 1. Scheme supported in *Mag*

The infrastructure architecture uses a JDBC connection to an external database which stores and retrieves specific information about the student, including student history and performance statistics.

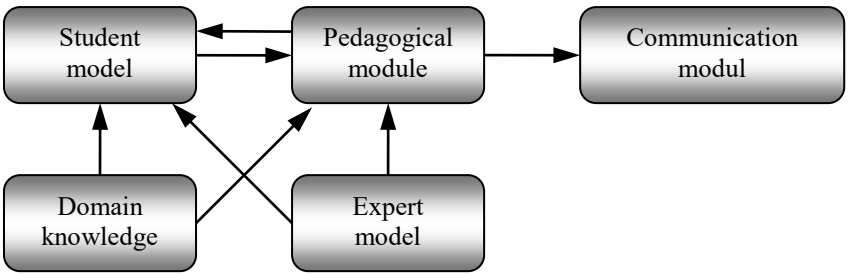

Fig. 2. Interaction of components in a tutoring system

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

Student Model. The student model stores information that is specific to each individual learner. At a minimum, such a model tracks how well a student is performing on the material being taught. A possible addition to this is to also record misconceptions. Since the purpose of the student model is to provide data for the pedagogical module of the system, all of the information gathered should be able to be used by the tutor.

Student model of a *Mag* system contains students' personal data as well as data about his progress in course. All necessary data are stored on server:

- student's personal data
- number and percentage of correct and incorrect answers for every particular test that student submits,
- data about problems in solving every particular types of question,
- student's grades for every particular lesson…

Using this data, administrator or system can personalize course and adopt it to every particular student by choosing the percentage of several different types of question that will be given to student.

Pedagogical Module. This component provides a model of the teaching process, by controling when to review, when to present a new topic, and which topic to present. As mentioned earlier, the student model should be used as input to this component, so the pedagogical decisions reflect the differing 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 will be presented to 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 tutor can access it.

Domain knowledge of *Mag* system contains collection of tutorials, examples and tests for 20 lessons. Lessons are grouped in several areas, and every lesson is supported with numerous examples and tests. Administrator is provided with opportunity for adding new lessons, examples and tests without any limit.

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 chosen by communication module. On the other hand, student can change the order in any time or choose what activity will be

taken next. Page for communication with mentor and detailed reviews of current progress are always available to student.

Expert Model. The expert model is similar to the domain. However, it is more than just a representation of the data; it is an appropriate approach for presenting the knowledge to student [6]. Most commonly, it is part of the system that is capable of solving problems from the domain. By using an expert model, the tutor can compare the student's solution to the solution, pinpointing the subject in which the learner had difficulties.

In many programming problems, there are often many possibilities for expert's solutions. An administrator may provide one solution to a problem but there may be many other solutions that are equally suitable. Expert model of *Mag* tries to evaluate students' answers rather than compare those to given solutions. First it is searching for syntax mistakes in the student's code and afterwards it is comparing results of executed program with the expected ones.

4.2 User Interface of *Mag*

Design of the student's user interface is a significant factor in designing computer-based programming tutors [9], therefore, it was given careful consideration during the design of *Mag*. *Mag* system has two major parts:

- application for administrators and
- Web pages for students.

Administrators' part gives the administrator the opportunity to get insight into students work with numerous reports, to change the course for every student particular, 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 a system. Web page that gives a student opportunity to test his knowledge is shown in figure 3. List of lessons' titles are shown in box list at the left side of interface. There is no predefined order for taking lessons, so student can skip current lesson at anytime. Every test contains several multiple-choice questions and problems with code completion, that 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.

5 Organization of Lessons and Testing the Students Knowledge

Whole course material is divided into learning objects. Learning objects are small units of learning, ranging from 2 to 15 minutes, according to SCORM (Sharable Courseware Object Reference Model), the ADL standards framework [7]. A LO is constructed from Media Assets, such as paragraphs of text or html, screen titles, captions, video, animation, diagrams, and sound narration [8].

In system *Mag*, learning objects are presented in the form of lessons. Every lesson contains three basic parts: tutorials, examples and tests. To every lesson unlimited number of examples and tests are attached. Their number could be even more increased by administrators.

Every tutorial contains explanation of concepts and appropriate syntax rules for material presented in the lesson. After tutorial, student is provided with several examples connected to the lesson. If student wants to exercise more examples he could choose *additional examples* option. At any stage of learning and processing particular lesson student can decide to start process of testing.

Most tutoring systems require the teacher to match problems with corresponding solutions. *Mag* is designed to provide the teachers an easy way to create the tutorials, examples and tests, and connect them.

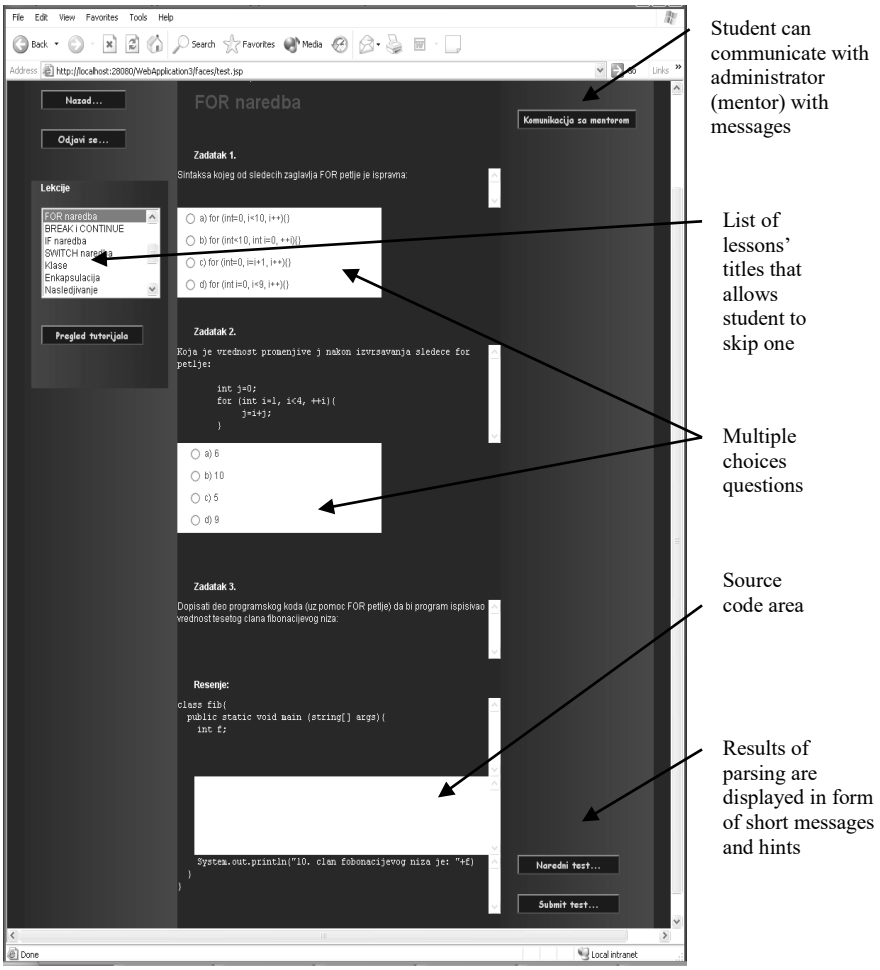

Fig. 3. *Mag*'s user interface

Tests connected to every lesson consist of two types of questions: multiple choice and code completion.

Multiple - choice question. This type of test is used to ask the learner to trace the correct sample code or to choose correct result after execution of code portion. Example of multiple choices question is shown in figure 4.

Code completion question. Problem is presented in a form of skeleton program with the specific area for entering code snippet according to program specification (figure 5).

For loop – test

Choose correct syntax of *for* loop:

```
a)  for (int=0, i<10, i++){}
b)  for (i<10, int i =0, ++i){}
c)  for (int i = 0, i=i+1, i++){}
d)  for (int i =0, i<9, i++){}
```

Fig. 4. Multiple choices question

After entering code and starting compilation, the learner can try to correct the syntax errors (in his code snippet) according to the error message returned. He can continue testing procedure if no errors are found.

Enter the code snippet to correctly complete Java program for computing 10th member of Fibonacci sequence

```
class fib{
  public static void main(string[] args){
    int f;
     /*student write code here*/

    System.out.println("10th memb is"+f);
  }
}
```

Fig. 5. Skeleton program

There is a number of appropriate hints for each incorrect response category, that will lead student to a correct answer. For example, student can be warned about missing key words, inappropriate syntax, wrong data types, etc.

Even though 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.

That is the reason for compiling and running complete program and presenting results to student.

Flow chart of *Mag* activities connected to testing of student's knowledge is shown in figure 6.

There is no predefined order for submitting answers in one test. If student has doubts about certain question, he can skip it and come back later. He could quit course in every moment and *Mag* will remember his current results and progress in his appropriate student model.

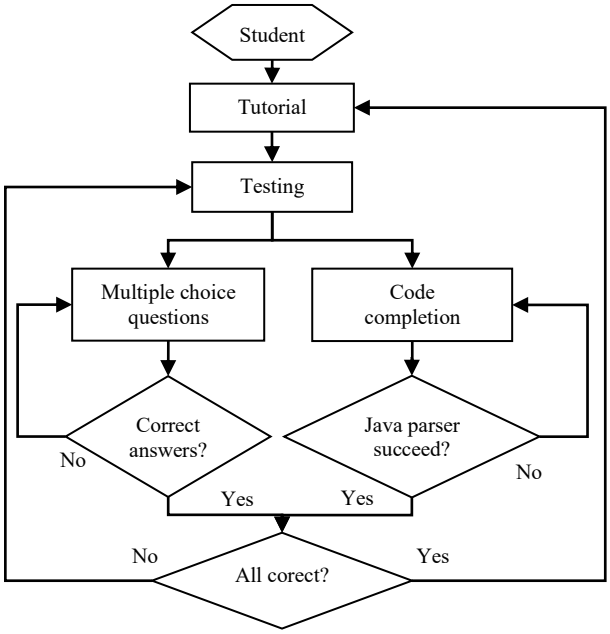

Fig. 6. Flow chart of *Mag* module for testing of student's knowledge

It was necessary to restrict the usage of *Mag* to small subset of the Java programming language due to the semantic complexity. The focus is put to the following list of Java language basics:

- variables declaration,
- operators and
- loop statements.

6 Conclusion

Mag is designed to combine distance education with traditional programming learning activities. It can provide student with a more convenient and efficient way to proceeding a distance Java programming course. When compared to other similar tools, *Mag* is more appropriate to nowadays needs. It provides optimal performance

with use of the most appropriate architecture and several improved tools for testing students knowledge and tools for guiding students towards answers.

This research is significant since it 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 that Java has become an extremely popular programming language in computer community.

References

1. J. R Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier, *Cognitive Tutors: Lessons learned,* The Journal of the Learning Sciences, 2002.
2. K. R. Koedinger, *Smart machines in education*. Cambridge, MA: MIT Press. 2001.
3. T. Murray, *Intelligent Tutoring Systems architecture,* Proceedings of the Third International Conference on Intelligent Tutoring Systems, Montreal, 1996.
4. http://pepite.univ-lemans.fr/
5. E. R. Sykes, F. Franek. *An Intelligent Tutoring System Prototype for Learning to Program Java*, The third IEEE International Conference on Advanced Learning Technologies (ICALT'03) Athens, Greece, 2003.
6. B. Woolf, *AI in Education. Encyclopedia of Artificial Intelligence*, John Wiley & Sons, Inc., New York 1995.
7. C. Shepherd, *E-Learning's Greatest Hits,* Above and Beyond, 2003.
8. A. Gallenson, J. Heins, T. Heins, *Macromedia MX: Creating Learning Objects*, Macromedia, Inc. 2002
9. R. Bednarik, A. Moreno, N. Myller, *Program Visualization for Programming Education – Case of Jeliot 3,* Department of Computer Science, University of Joensuu, 2006
10. F. Franek, E. R. Sykes, *Inside the Java Intelligent Tutoring System Prototype: Parsing Student Code Submissions with Intent Recognition*, The Journal of the Learning Sciences, 2005
11. G. Blank, S. Parvez, F. Wei, S. Fang, *A Web-based ITS for OO Design,* Computers in Education, 2005
12. B. Vesin, M. Ivanovic, Z. Budimac, & I. Pribela, Mile-Multifunctional Integrated Learning Environment. In e-Learning, 2007. pp. 104-108.
13. B. Vesin, & M. Ivanović, Modern educational tools. In Proceedings of PRIM2004, 16th Conference on Applied Mathematics, Budva, Montenegro, 2004. pp. 293-302.