Different Roles of Agents in Personalized Programming Learning Environment

Mirjana Ivanović[1], Dejan Mitrović[1], Zoran Budimac[1], Boban Vesin[2], and Ljubomir Jerinić[1]

[1] Department of Mathematics and Informatics
Faculty of Sciences, University of Novi Sad, Serbia
{mira,dejan,zjb}@dmi.uns.ac.rs, jerinicl@eunet.rs
[2] Higher School of Professional Business Studies
University of Novi Sad, Serbia
bobanvesin@yahoo.com

Abstract. Researchers had recently begun to investigate various techniques to help learners to improve learning effects using e-learning systems. In this paper, we propose an e-learning architecture with a recommendation module consisting of several different kinds of pedagogical agents which actively participate in learning processes, provoking learners and motivating them to learn more effectively. Different kinds of agents can be introduced in order to support scaffolding activities in learning programming languages and problem solving.

Keywords: e-learning, harvester agents, pedagogical agents, content personalization.

1 Introduction

Recently, in different learning environments there has been a trend of using adaptive and intelligent web-based educational systems. These systems can use different recommendation and scaffolding techniques in order to increase motivation for learning and to suggest online learning activities or optimal browsing pathways to learners, based on their different preferences and levels of knowledge. Their main objective is to adapt and personalize learning to the needs of a learner as much as possible [33].

The task of delivering personalized content is often framed in terms of a recommendation so some recommender systems have been applied in e-Learning environments for recommending objects or concepts that learners should study next [10].

Another approach in designing and developing educational tools include employment of different kinds of agents. For the purpose of our research we distinguish two groups of agents. The first group consists of *pedagogical* agents with diverse functionalities and potential. The second group includes *harvesting* agents employed in educational environments with the main goal of collecting different learning resources.

D.K.W. Chiu et al. (Eds.): ICWL 2011/2012 Workshops, LNCS 7697, pp. 161–170, 2014.

Pedagogical agents have been defined as "lifelike characters presented on a computer screen that guide users through multimedia learning environments" [13]. Their main design goal is to engage, motivate, and guide learners through the learning process. According to [14], one of the main characteristics of pedagogical agents is interactivity. That is, they not only provide answers and additional learning material, but also generate questions and propose solutions. Our proposed architecture is based on two types of pedagogical agents, described later.

In general, a *harvester* is an object that collects resources and metadata records from remote repositories. The main advantage of using agents as harvesters is parallel execution of federated searches in heterogeneous repositories [8]. Additional benefits come from inter-agent communication and mobility, which, respectively, provide the means for exchanging the harvested material and reducing the load imposed on the central repository. Recently, our research has been concentrated on exploring various ways of using harvesting agents in order to improve learning quality in our tutoring system [15]. The proposed architecture is designed to employ software agents that dynamically recognize and propose possible learning material as additional learning objects to learners. In the future, our intention is to combine various harvested learning components into highly-personalized additional teaching material.

In this paper initial ideas on how to incorporate different kinds of agents – *Harvesting*, *Provoking*, and *Zestful* – into an existing personalized learning and tutoring environment will be presented. The role of each proposed agent is adjusted in order to increase learners' motivation and improve the quality of learning, increasing in such a way the amount of acquired knowledge. In particular, we are concentrating on learning programming languages and solving programming tasks. For our ideas we have used an improved web-based learning system architecture presented in [19]. The proposed architecture, which provides adaptive courses using hypermedia and recommendations, rose from an existing web-based Java tutoring system *Mag* [32].

The rest of the paper is organized as follows. In Section 2, an overview of work related to our research is given. Section 3 outlines the vital functionalities of the web-based e-learning architecture, the starting point for our current research. Section 4 describes *MagMAS*, our proposition of the e-learning architecture which incorporates three kinds of agents. Section 5 concludes the paper and outlines possible future work.

2 Related work

As our primary goal is to incorporate different kinds/roles of agents in an existing personalized learning environment for programming languages, in this section we will shortly discuss agent-oriented learning systems that can help us to distinguish and tailor our own agents and their particular roles.

A critical analysis of e-learning with hypertext and hypermedia presented in [1] indicates that the usage of this form of learning material can be very challenging for learners. Hypermedia learning environments should, therefore, incorporate scaffolding in order to help learners in overcoming the obstacles imposed by complex and challenging topic. The introduction of agent-based support for scaffolding activities is

one of the primary goals of our approach and the proposed architecture incorporates two kinds of scaffolding agents.

The majority of tutoring systems for learning programming languages are just well-formatted versions of lecture notes or textbooks. Some of these systems *JavaBugs* [27] and *JITS* [29, 30], are used for learner assessment, while others like *Jeliot 3* and *Logic-ITA* [3, 17], offer basic tutoring. One of the interesting systems for our research is *Java Intelligent Tutoring System* (*JITS*) [30]. *JITS* implements *Java Error Correction Algorithm* (*JECA*), that enables compilers to identify source code errors more clearly, and to intelligently fix the code accordingly. In our approach some of these functionalities will be included in two kinds of agents (*Provoking* and *Zestful*). *Jeliot 3* is a program animation tool that displays graphically the execution of Java-based object-oriented programs. In the future work, a similar functionality will be added along with our pedagogical agents.

Agents are used as means of extending intelligent tutoring and course recommendation in different systems: *Educ-MAS* [11], *MathTuthor* [5], and *ABITS* [4]. Instead of using harvesting agents, however, these systems usually offer metadata authoring tools. Some of these concepts, combined with our own experience in developing educational environments [16, 19, 23] are incorporated in our proposed agent architecture which includes *Harvesting*, *Provoking* and *Zestful* agents.

Agent Based Search System (*ABSS*) [22] and the *AgCAT* system [2] represent more complete agent-based frameworks for harvesting learning objects. *ABSS* seems to be more powerful as it uses *Personal Agents* to offer personalized search capabilities, customized presentation of the available material, etc. The general architecture which encompasses harvesting agents used in our approach [15] is similar to *ABSS* and *AgCAT*. However, while metadata harvesting agents in *ABSS* and *AgCAT* constantly monitor the changes in remote repositories, harvesting agents in our system are dispatched to remote LO repositories in response to an automatically detected decline in learner's performance. Additionally our system includes agents that monitor the learner's progress through a course, and obtain new learning material as needed.

In order to assess motivational and learning impact pedagogical agents have on learners, an in-depth analysis of 39 studies has been presented in [13]. The initial conclusion was discouraging – only 5 studies have reported advantages of using agents on learning, while in only 1 study a positive impact of pedagogical agents on the motivation has been recorded. However, after further analysis it was noted that only 15 studies featured a control group (i.e. without an agent), while only 4 out of these 15 applied motivational measures. Therefore, the proposal is for researchers to focus on examining the circumstances (e.g. the environment, domains, levels of design, etc.) under which the use of agents yields in an improved learning outcome.

One of the important factors of the effects of pedagogical agents outlined in [13] is their graphical representation. It was observed that the use of a pedagogical agent can have negative effects on the learning outcome, if learners dislike the agent's visual appearance. Similarly, in [12] visual clues are defined as important factors people rely on in order to form expectations for guidance and interaction.

A successful usage of an animated pedagogical agent was reported in [28]. The agent, a lifelike character with gesture support, was incorporated into an existing

intelligent tutoring system *SQL-Tutor* for learning *SQL*. A study was then carried out, showing that learners who have been using the agent found the *SQL-Tutor* system to be "more enjoyable and helpful." When compared to our proposed pedagogical agents, the agent used with *SQL-Tutor* is relatively simple, and was employed just as a more pleasant way of presenting the learning material.

Obviously, there is a lot of ongoing research in the field of agent-based e-Learning systems. Although existing systems do include sophisticated metadata harvesting approaches, they do not seem to incorporate agents in both the metadata harvesting, and the course recommendation levels. This is the goal of our proposed architecture.

3 Architecture of the Existing Tutoring System

The architecture of the e-learning/tutoring system which represents the starting point for our agent-oriented approach is an extension of an existing web-based Java tutoring system *Mag* [16] that has been developed at our Department. *Mag* is system designed to help learners in studying programming languages during different courses [31]. It is an interactive system that allows learners to use teaching material prepared for programming languages and to test the acquired knowledge within appropriate courses. *Mag* is a multifunctional educational system that fulfills three primary goals [18]:

- Provide a tutoring system for learners in a platform-independent manner
- Provide teachers with useful reports that identify the strengths and weaknesses of the learner's learning process
- Provides a rapid development tool for creating basic elements of tutoring systems: new learning objects, units, tutorials, and tests

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

Different techniques must be implemented to adapt content delivery to individual learners in accordance to their learning characteristics, preferences, styles, and goals. In order to support adaptive learning and personalization of the content delivery, the system administrators must constantly measure the learner's knowledge and progress, build learner model, and redirect the course as needed. The first version of *Mag* was further extended with an introduction of two general categories of personalization:

- *Content adaptation* – presenting the content in different ways, according to the domain model and information obtained from the learner model.
- *Link adaptation* – the system modifies the appearance and/or availability of every link that appears in a course web page, in order to show to the learner whether the link leads to interesting new information, to new information the learner is not yet ready for, or to a page that provides no new knowledge.

Open standards, like XML, RDF and OWL [20, 24] needed to be used in order to allow the specification of ontologies to standardize and formalize meaning and to

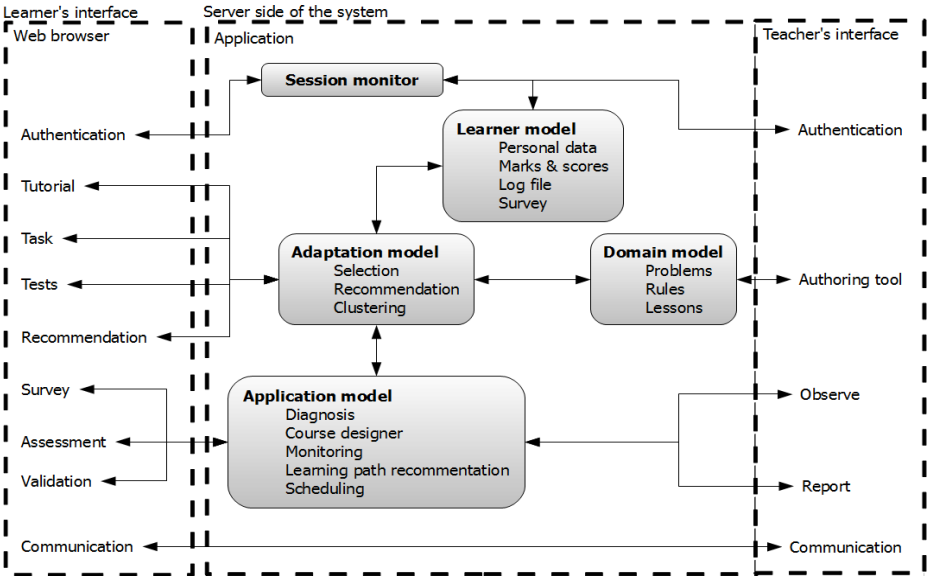

Fig. 1. Web-Based Learning Architecture

enable the reuse and interoperability. Fig. 1 shows the adapted architecture of the first version of *Mag* [31], which was developed on the basis of experiences with similar web-based learning systems [6, 20, 25], as well as ontology-supported adaptive web-based education systems [7, 9].

The core of the system includes *adaptation model*, *learner model*, *application model*, and *domain model*. The domain model represents storage for all essential concepts, tutorials, and tests in the domain. The whole course is divided into units which are all consisted of several lessons. Each lesson consists of three basic parts: tutorials, programs/examples connected to tutorials, and tests (an unlimited number of examples and tests can be attached). At the end of each lesson a post-test is conducted. A test contains several multiple-choice questions and code completion tasks that learners have to solve and present the level of acquired knowledge. *Provoking* and *Zestful* agents have to play essential role in this process. The application model applies different strategies/techniques to ensure efficient tailoring of the learning content and personalized tasks to individual learners. It supports the given pedagogical strategy.

The adaptation model follows the instructional directions specified by the application model and creates navigation sequence of resources recommended for the particular learner. Therefore the idea is to incorporate *Provoking* and *Zestful* agents in the adaptation model (more details are presented in next section). These two components are separated in order to provide an easier addition of new content clusters and adaptation of functionalities.

Each learner model is a collection of both static (personal data, specific course objectives, etc.) and dynamic (marks, scores, time spent on specific lesson, etc.) data about the learner, as well as a representation of learner's performances and learning history. Within the session monitor component, the system gradually builds the

learner model during each session, in order to keep track of the learner's actions and his/her progress, detect and correct errors and possibly redirect the session as needed. The adaptation model is also responsible for building and updating characteristics of the learner model and for personalization of the application to the learner. It processes the changes in the learner's behavior (e.g. learner's activities) and provides an adaptation of visible aspects of the system accordingly. Its main tasks also include the storage and management of course data, various ways of presenting courses to learners, provision of reports and test results etc.

Educational ontologies for different purposes must be included, i.e. for presenting a domain (domain ontologies), for building the learner model (learner model ontologies), or for presenting activities in the system (task ontologies) [25]. A repository of ontologies must be built in order to achieve easier knowledge sharing and reuse, more effective learner modeling, and easier extension of the system.

4 *MagMAS* – An Agent-Oriented e-Learning Architecture

In this section we will present main ideas and functionalities of the new agent-oriented e-learning architecture named *MagMAS*, which is an upgraded version of the existing web-based Java tutoring system *Mag* [16].

In the course of further improvements of the original *Mag*, three different levels of personalization (based on levels of increasing abstraction and sophistication) that must be included in the *Mag* system [19] have been suggested in [9, 26]: *Self-Described Personalization*, *Segmented Personalization*, and *Cognitive-Based Personalization*.

Research presented in this paper is concentrated on the *Segmented Personalization* style of learning, with the following characteristics: learners will be grouped into smaller, identifiable and manageable clusters, based on their common attributes (e.g. class and age), preferences, and survey results. Parts of the instructions are then tailored to these groups, and are applied in the same or similar way to all members within a single group. Learning material will also be clustered by its purpose, based on the benefits it delivers to learners. In spite of the fact that the *Mag* system supports three types of tasks/questions [16] – *Multiple-choice of syntax check*, *Multiple-choice of execution results*, and *Code completion* – the focus of the research in this paper will be on the last type.

Within the *Code completion* type of questions, a problem is presented in form of a skeleton program with parts of the code missing. The learner is expected to enter the appropriate code snippet according to the program specification. Code completion tasks are used to check the learner's programming skills: if he/she has difficulties with a particular kind of questions or tasks, the system will increase their number in the next session in order to provide the learner with additional opportunities for improving the skills in mention.

The contribution of this paper is outlined in Fig. 2. It shows how application and domain models of the original *Mag* system were updated to include *Provoking* and *Zestful* pedagogical agents.

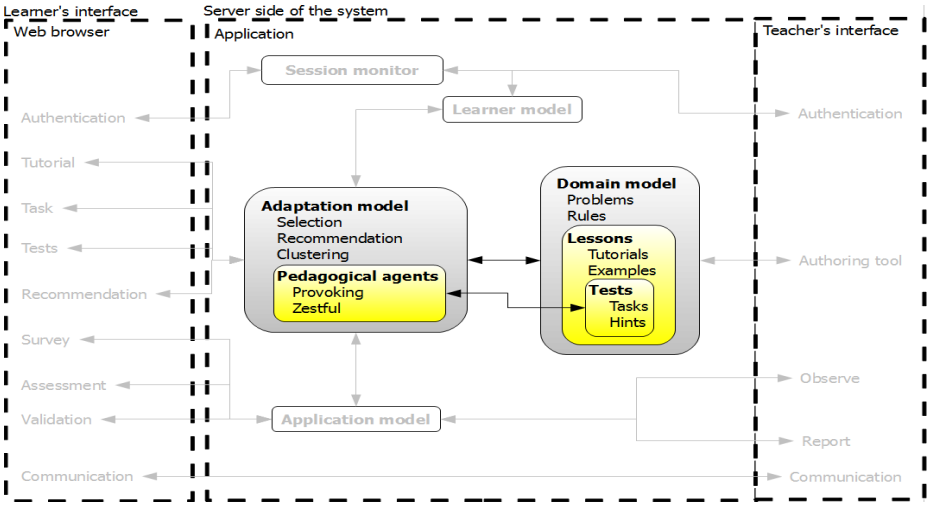

Fig. 2. Architecture of the proposed *MagMAS* agent-based e-learning system

As already mentioned, to each lesson an arbitrary number of code completion tasks is attached. At the end of the lesson, the learner is required to conduct a post-test offered by the system. It includes mainly code completion tasks that evaluate the acquired programming knowledge and skills. Newly introduced *Provoking* and *Zestful* agents play an essential role in this process. They try to motivate the learner to correctly apply the acquired programming knowledge that was presented to him/her during the appropriate tutorial, and also illustrated by several accompanying examples.

After a completed lesson, the set of tasks to be accomplished by the learner will be chosen intelligently, based on the stored learner model, but also on the learner's immediate reactions to agents' proposals. For each programming task, a numerous *hints* are prepared (by teacher, in this version of system) and attached to code completion tasks in the domain model. Hints are prepared in two forms: the first form includes *useful* hints which will be used by *Zestful* agent, while the second form includes *provoking/somehow wrong* hints which will be used by *Provoking* agent.

Both *Zestful* and *Provoking* agents have the same goal – to increase the learner's quality of learning – but approach to this task differently. The *Zestful* agent offers hints that, in the usual methodology of teaching programming languages, represent useful directions for problem solving as propositions of appropriate solutions to the given problem. The *Provoking* agent, on the other hand, tries to steer the process of learning and problem solving in a bad direction by offering wrong parts of the code. That is, it offers false hints and suggests bad solutions to the given problem. The basic rationale behind this approach is to encourage learners not to follow the tutor's instructions blindly, but rather to employ critical thinking and, in the end, they themselves decide on the proper solution to the problem.

In the rest of the section we will shortly present an example of how our agents act in order to direct the learner. The task is to complete a Java program for calculating

the first 10 members of the Fibonacci sequence. The problem is presented in a form of skeleton program with a specific area for entering the code snippet. The teacher has prepared several positive and provoking hints, which the agents use to stimulate the learner to reach the correct solution using the `for` statement. Some possible hints/appropriate sentences agents will use during conversation are:

Zestful Agent

1. `for (int i = ?; i < 10; i++){}` – "What should be the starting index? Remember that the first element of the Fibonacci sequence has the index 0, while the expression for calculating other elements is $f_i = f_{i-1} + f_{i-2}$"
2. `for (int i = 0; i <= ?; i++){}` – "What should be the ending index? Although you need 10 numbers, remember that the index of the first element is 0."
3. `for (int i = 0; i < 10; ?){}` – "Should you use `++i` or `i++` to modify the value of `i`? Remember that this modification is always executed *at the end* of the `for` loop."

Provoking Agent

1. `for (int i = ?; i < 10; i++){}` – "What should be the starting index? Hint: the first element of the Fibonacci sequence is often denoted as f_0"
2. `for (int i = 0; i <= ?; i++){}` – "What should be the ending index? Hint: look at the initialization of the array `f` – how many elements does it have?"
3. `for (int i = 0; i < 10; ?){}` – "Should you use `++i` or `i++` to modify the value of `i`? Remember that `++i` first increases the value of `i`, and then uses the new value in an expression."

5 Conclusions and Future Work

During the last couple of years we have been conducting development and implementation of several different learning tools and architectures, starting from traditional approaches [16, 23, 32] and towards the incorporation of recent approaches like recommender systems, personalized content delivery [19], usage of agents [15]. Our first attempt resulted in incorporating harvesting agents that facilitate process of obtaining new/additional learning material for existing course material. In this paper we went a step ahead as our primary goal was to include different kinds of pedagogical agents in the environment for learning programming languages. Introduced pedagogical (*Zestful* and *Provoking*) agents actively participate in learning processes, provoking and motivating learners to learn more effectively.

Of course, these initial ideas have to be further improved and practically implemented and evaluated. Additional efforts have to be put into obtaining more sophisticated useful and provoking hints, nevertheless they are prepared in advance by teacher or harvested from different learning repositories.

On the other hand there are some additional rooms for improvements. For example *Moodle* [21] system has been used within the majority of computer courses at our Department. The learners have been accustomed to using *Moodle* as the fundamental

tool for learning activities. The client front-end of the system could be therefore designed as a *Moodle* plug-in. In this way, we will be able to seamlessly introduce the system to learners, i.e. in a, for them, familiar learning environment.

Finally some further improvements of the system could be directed in the domain of code visualization and animation. Both kinds of agents introduced in *MagMAS* could be enhanced by this functionality, having in mind tools like *Jeliot 3*.

Acknowledgements. The work is partially supported by Ministry of Education, Science and Technological Development of the Republic of Serbia, through project no. OI174023: "Intelligent techniques and their integration into wide-spectrum decision support".

References

1. Azevedo, R., Jacobson, M.J.: Advances in scaffolding learning with hypertext and hypermedia: a summary and critical analysis. Education Tech. Research Dev. 56, 93–100 (2008)
2. Barcelos, C.F., Gluz, J.C.: An agent-based federated learning object search service. Interdisciplinary Journal of E-Learning and Learning Objects 7, 37–54 (2011)
3. Bednarik, R., Moreno, A., Myller, N.: Program Visualization for Programming Education – Case of Jeliot 3. ACM New Zealand Bulletin, P. 8 (2006)
4. Capuano, N., Marsella, M., Salerno, S.: ABITS: an agent based intelligent tutoring system for distance learning. In: Proceedings of the International Workshop in Adaptative and Intelligent Web-Based Educational Systems, pp. 17–28 (2000)
5. Cardoso, J., Guilherme, B., Frigo, L.B., Pozzebon, E.: MathTutor: a multi-agent intelligent tutoring system. In: First IFIP Conference on AIAI, pp. 231–242 (2004)
6. Chen, C.M.: Ontology-based concept map for planning a personalized learning path. British Journal of Educational Technology, 1–31 (2008)
7. De Bra, P., Aroyo, L., Chepegin, V.: The next big thing: adaptive Web-based systems. Journal of Digital Information 5, Article No. 247, 12 (2004)
8. De la Prieta, F., Gil, A.B.: A multi-agent system that searches for learning objects in heterogeneous repositories. In: PAAMS Special Sessions and Workshops, vol. 71, pp. 355–362 (2010)
9. Devedžić, V.: Education and the Semantic Web. International Journal of Artificial Intelligence in Education 14, 39–65 (2004)
10. Farzan, R., Brusilovsky, P.: Social Navigation Support in a Course Recommendation System. In: Proceedings of 4th International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Dublin, pp. 91–100 (2006)
11. Gago, I.S.B., Werneck, V.M.B., Costa, R.M.: Modeling an educational multi-agent system in maSE. In: Liu, J., Wu, J., Yao, Y., Nishida, T. (eds.) AMT 2009. LNCS, vol. 5820, pp. 335–346. Springer, Heidelberg (2009)
12. Haake, M., Haake, G.A.: Visual stereotypes and virtual pedagogical agents. Educational Technology & Society 11(4), 1–15 (2008)
13. Heidig, S., Clarebout, G.: Do pedagogical agents make a difference to student motivation and learning? Educational Research Review 6, 27–54 (2011)
14. Heller, B., Procter, M.: Animated pedagogical agents and immersive worlds: two worlds colliding. Emerging Technologies in Distance Education, pp. 301–316. Athabasca University Press (2010)

15. Ivanović, M., Mitrović, D., Budimac, Z., Vidaković, M.: Metadata harvesting learning resources – an agent-oriented approach. In: 15th International Conference on System Theory, Control and Computing, ICSTCC 2011, pp. 306–311 (2011)
16. Ivanović, M., Pribela, I., Vesin, B., Budimac, Z.: Multifunctional Environment for E-Learning Purposes. Novi Sad Journal of Mathematics 38(2), 153–170 (2008)
17. Jeliot 3 homepage, http://cs.joensuu.fi/jeliot/
18. Jones, N., Macasek, M., Walonoski, J., Rasmussen, K., Heffernan, N.: Common Tutor Object Platform – an e-Learning Software Development Strategy. In: Proceedings of the 15th International Conference on World Wide Web, Edinburgh, Scotland, pp. 307–316 (2006)
19. Klašnja-Milićević, A., Vesin, B., Ivanović, M., Budimac, Z.: Integration of Recommendations and Adaptive Hypermedia into Java Tutoring System. ComSIS Journal 8(1), 211–224 (2011)
20. Muñoz Merino, P.J., Kloos, C.D.: An Architecture for Combining Semantic Web Techniques with Intelligent Tutoring Systems. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) ITS 2008. LNCS, vol. 5091, pp. 540–550. Springer, Heidelberg (2008)
21. Moodle homepage, http://moodle.org/
22. Orzechowski, T.: The use of multi-agents' systems in e-learning platforms. Siberian conference on control and communications. In: SIBCON 2007, pp. 64–71 (2007)
23. Pribela, I., Ivanović, M., Budimac, Z.: Svetovid – interactive development and submission system with prevention of academic collusion in computer programming. British Journal of Educational Technology 40(6), 1076–1093 (2009)
24. SCORM homepage, http://www.adlnet.gov/scorm/
25. Šimić, G.: The Multi-courses Tutoring System Design. ComSIS 1(1), 141–155 (2004)
26. Soonthornphisaj, N., Rojsattarat, E., Yim-ngam, S.: Smart E-Learning Using Recommender System. In: Huang, D.-S., Li, K., Irwin, G.W. (eds.) ICIC 2006. LNCS (LNAI), vol. 4114, pp. 518–523. Springer, Heidelberg (2006)
27. Suarez, M., Sison, R.C.: Automatic Construction of a Bug Library for Object-Oriented Novice Java Programmer Errors. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) ITS 2008. LNCS, vol. 5091, pp. 184–193. Springer, Heidelberg (2008)
28. Suraweera, P.: An animated pedagogical agent for SQL-Tutor. Honours Report (1999)
29. Sykes, E.R., Franek, F.: An Intelligent Tutoring System Prototype for Learning to Program Java. In: The 3rd IEEE ICALT 2003, Athens, Greece, pp. 485–492 (2003)
30. Sykes, E.R., Franek, F.: Presenting JECA: A Java Error Correcting Algorithm for the Java Intelligent Tutoring System. In: Advances in Computer Science and Technology - 2004, St. Thomas, US Virgin Islands (2004)
31. Vesin, B., Ivanović, M., Budimac, Z., Pribela, I.: Tutoring System for Distance Learning of Java Programming Language. In: 10th Symposium on Programming Languages and Software Tools SPLST, Dobogókő, Hungary, pp. 310–320 (2007)
32. Vesin, B., Ivanović, M., Budimac, Z.: Learning Management System for Programming in Java. Annales Universitatis Scientiarum De Rolando Eötvös Nominatae, Sectio - Computatorica 31, 75–92 (2009)
33. Zaiane, O.R.: Recommender systems for e-learning: toward non-intrusive web mining. In: Data Mining in E-Learning, pp. 79–96. WIT Press (2006)