INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 10(2):200-210, April, 2015.

HAPA: Harvester and Pedagogical Agents in E-learning Environments

M. Ivanović, D. Mitrović, Z. Budimac, L. Jerinić, C. Bădică

Mirjana Ivanović*, Dejan Mitrović, Zoran Budimac, Ljubomir Jerinić
Department of Mathematics and Informatics
Faculty of Sciences, University of Novi Sad, Serbia
mira@dmi.uns.ac.rs, dejan@dmi.uns.ac.rs, zjb@dmi.uns.ac.rs, jerinic@dmi.uns.ac.rs
*Corresponding author: mira@dmi.uns.ac.rs

Costin Bădică
Computer and Information Technology Department
Faculty of Automatics, Computers and Electronics,
University of Craiova, Romania
cbadica@software.ucv.ro

Abstract: In the field of e-learning and tutoring systems two categories of software agents are of the special interest: *harvester* and *pedagogical* agents. This paper proposes a novel e-learning system that successfully combines both of these agent categories and introduces two distinct sub-types of pedagogical agents *helpful* and *misleading*. Whereas helpful agents provide the correct guidance for the given problem, misleading agents try to guide the learning process in the wrong direction by offering false hints and inadequate solutions. The rationale behind this approach is to motivate students not to trust the agent's instructions blindly, but to employ critical thinking. Consequently, students will be put in a "softly stressed" environment in order to prepare them for real working environments in their future work in companies. Nevertheless students themselves will decide on the correct solution to the problem in question.
Keywords: E-learning, adaptability, personalization, intelligent agent, harvester agents, pedagogical agents.

1 Introduction

Software agents (or simply *agents*), can be defined as *autonomous* software entities with various degrees of *intelligence*, capable of exhibiting both *reactive* and *pro-active* behavior in order to satisfy their design goals. From the point of e-learning and tutoring systems, two types of agents are of special research interest: *harvester* and *pedagogical*. Harvester agents collect learning material from online, heterogeneous repositories. The core properties of the agent technology (e.g. parallel and distributed execution, mobility, and inter-agent communication) can bring significant benefits to the harvesting process [16]. Pedagogical agents can be defined as "lifelike characters presented on a computer screen that guide users through multimedia learning environments" [6]. Their main goals are to motivate and guide students through the learning process [7].

This paper presents a stand-alone e-learning architecture named *HArvester and Pedagogical Agent-based e-learning system* (HAPA), and designed to help learners during solving programming tasks. HAPA consists of three main components: *Harvester agents*, *Classifier module*, and *Pedagogical agents.* The harvester agents collect the appropriate learning material from the web. Their results are fed into the Classifier module, which performs automatic classification of individual learning objects. Finally, a pair of specially designed Pedagogical agents - one *helpful* and one *misleading* - is used to interact with students and guide them to comprehend the learning material. The helpful pedagogical agent provides useful hints for the problem in question. The misleading pedagogical agent guides the learning process in the wrong direction. Because the student is never sure with which agent (s)he is interacting, this novel approach encourages students not to follow the agent's/tutor's instructions blindly, but rather to employ critical thinking. We believe that this "softly stressed" environment could help learners to face stressful and competitive real working environments.

To the best of our knowledge, none of the existing e-learning systems employs this combination of pedagogical agents in conjunction with harvester agents for collecting additional learning material. This is the main idea and contribution behind the work presented in this paper. The initial ideas of using harvester and two types of pedagogical agents were presented in [8,9]. This paper concentrates on improvements and a concrete implementation of initial ideas, with a well-defined set of functional components. Harvester agents are now defined as *web crawlers* [11], specialized for collecting Java source code examples. The new Classifier module has also been defined, and a set of tools for preparing helpful and misleading hints, and visual representation of pedagogical agents has been implemented. Early evaluation results are presented as well.

The rest of this paper is organized as follows. Section 2 provides an overview of the existing work related to the employment of harvester and pedagogical agents. In Section 3, a detailed insight into the proposed system is presented. Section 4 brings implementation details and early evaluation results. Overall conclusions are given in Section 5.

2 Related work

There are many interesting approaches to using software agents in e-learning and tutoring environments. For example, *ABITS* [2], *MathTuthor* [3], and *Educ-MAS* [4] incorporate intelligent agents in order to improve the students' learning outcomes. However, none of these systems use harvester and pedagogical agents in the same environment.

It was shown in [17] that inherited properties of the agent technology – parallel and distributed execution – can be used to optimize the web crawling process of harvesting agents. The same approach has been taken in HAPA, except that our agents are highly specialized to search for syntactically correct Java source code examples.

Agent Based Search System (ABSS) relies on harvester agents to improve the quality of search query results [15]. The system is capable of not only harvesting heterogeneous remote learning object repositories, but also tracking changes in them. Similarly, *AgCAT* represents an agent-based federated catalog of learning objects [1]. The harvesting process is delegated between two agents - *Librarian* and *InterLibrarian* - that, respectively, maintain the local repository of learning objects, and perform the federated search and retrieval on remote repositories.

Both of these systems use sophisticated harvesting agents to retrieve the best-suited learning objects. The difference between HAPA and both ABSS and AgCAT is in the approach used to deliver the harvested content. ABSS and AgCAT are sophisticated search engines; they enable their users to *pull* the data using search queries. On the other hand, our system monitors and evaluates the student's progress through a course. If a decline in student's performance is detected, HAPA can harvest additional appropriate learning material and then *push* it to the student.

An interesting analysis of 39 studies related to the effects of pedagogical agents onto the learning outcome has been presented in [6]. The initial conclusion is that only 5 studies have detected positive effects of using pedagogical agents. However after a more detailed analysis it was observed that only 15 of the 39 studies used a control group without an agent, while actual motivational approaches were implemented in only 4 of these 15 studies.

Our first intention was not to implement visual representations of pedagogical agents in HAPA environment. However since several studies [5,6] discuss that un-appealing visual representations of an agent can have a negative impact on the student's willingness to interact with the agent, we decided to visualize our pedagogical agent(s). We implemented a simple character (see Figure 2) and let students decide to use it or not during learning and assessment activities. Both versions of our pedagogical agents are represented with the same visual character.

SmartEgg is a web-based pedagogical agent that assists students in learning SQL [13]. It is integrated into an intelligent e-learning system *SQL-Tutor*. The agent includes a visual representation with animated gestures, and can express different behaviors: introductory, explanatory, and congratulatory. SmartEgg is relatively simple, and is employed just as a more pleasant way of presenting the learning material.

As shown, there is a lot of ongoing research related to the usage of harvester and pedagogical agents in e-learning environments. However, although many existing systems incorporate either type of agents, there have been no previous attempts to efficiently integrate both harvesting and motivational-level agents. An additional, and more important contribution of HAPA is the concept of helpful and misleading pedagogical agents.

3 HAPA System overview

HAPA is currently a stand alone e-learning system which helps students in learning and especially in solving programming problems. At a later stage HAPA could be included as a component in other tutoring systems devoted to learning programming languages. Our intention is to incorporate it in *Protus*, a tutoring system we designed to help learners in learning essentials of Java programming language [10, 19].

A high-level overview of the system architecture is outlined in Figure 1. HAPA includes several important components: harvester agents, the Classifier module, repositories of helpful and misleading hints, and pedagogical agents. The functioning of each component and their mutual interactions are described in more details in the following sub-sections.

3.1 Harvesting and classifying the learning material

HAPA is mainly focused on the *code completion* type of tasks, in which students are expected to fill-in missing parts of the program. The student is given a code snippet, and then requested to complete the source code to meet the program specification. Code completion tasks are well-suited for both testing and improving the student's programming skills, because they require a thorough understanding of the underlying programming concepts.

As the initial step of constructing the code completion tasks, the additional learning material is collected by the harvester agents. The learning material consists of Java source code examples. With the abundance of these examples available on the web, harvester agents have been implemented as web crawlers [11]. The system administrator (e.g. the teacher) specifies starting web pages. Harvester agents scan these pages in search of syntactically valid Java source code examples. First, a simple search for some more important Java keywords, such as *class*, is performed. A block of text that contains keywords is then processed by a syntax analyzer to determine whether it is a valid Java program. The text can either be directly embedded in the web page or attached to the page as an external file (e.g. a *ZIP* archive).

After it processes the current page, the agent continues the harvesting process on all pages linked from the current one, and so on. Many agents can be deployed on a computer cluster and perform the harvesting in parallel. A centralized repository of visited pages is maintained in order to avoid duplicate work. The harvested learning material is fed into the Classifier module (see Figure 1), which automatically associates each Java source code example with a concrete lecture topic. The classification is performed via the static source code analysis. The Classifier module

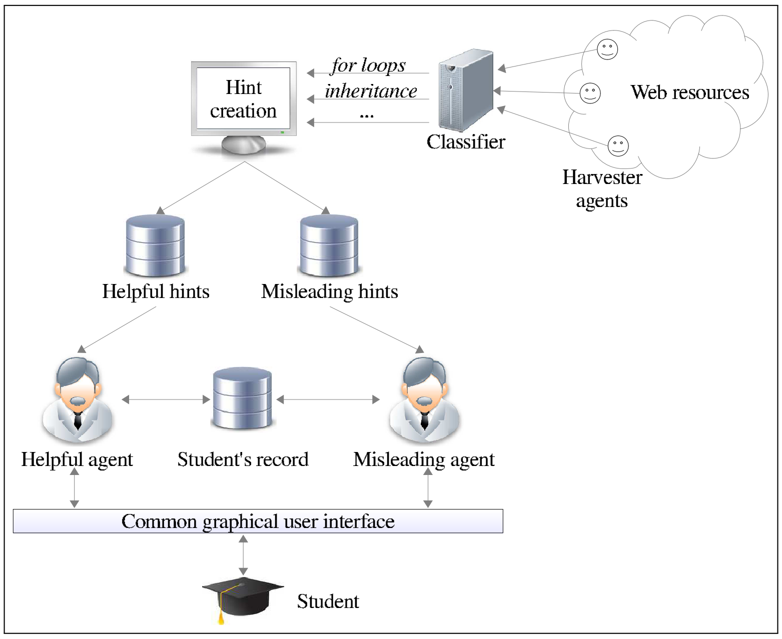


Figure 1: A high-level overview of the HAPA system.

constructs an *abstract syntax tree* for the given Java example, and then inspects programming constructs that appear in the tree. As a result, each example is assigned to the appropriate lecture topic. In return, the teacher is able to analyze and focus on examples of a particular interest, i.e. those that are directly attached to the lecture topic in question.

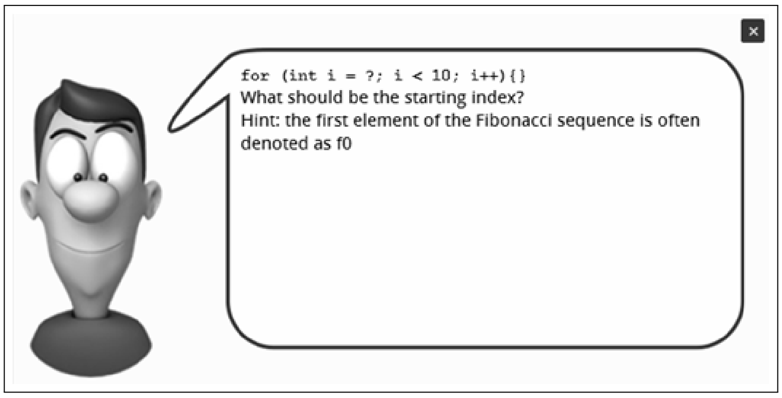
Currently, the Classifier's decision on which example belongs to which topic is a suggestion to the teacher. In the end, the teacher makes the final selection and filter the obtained source code examples. In order to improve the Classifier's performance, more intelligent source code classification techniques will be implemented in the future (e.g. [12]).

Once the harvested learning material has been classified, the teacher can use them select most appropriate solutions for learning topic and prepare the code completion tasks. This step is performed manually, using a specially designed GUI tool. The tool enables teacher to quickly scroll through the classified Java source code examples, select the ones to be actually used, and process them by removing parts of the code and constructing useful and misleading hints which will be offered to students. The hints are incorporated in pedagogical agent and used in learning.

3.2 Pedagogical agents

The significant novelty of this work is incorporation in a learning environment two different types of pedagogical agents – helpful, and misleading. Both agents are hidden from the student behind the same interface and visual representation (see Figure 2), and take turns in interacting with the student at random time intervals. Therefore, the student is never sure with which agent he/she is interacting. The rationale behind this approach is to motivate students not to trust the agent's hints blindly. Instead, they should critically analyze the problem and the proposed hint, and independently decide on the proper solution.

In the scientific literature and the actual software products, it is common to represent pedagogical agents as lifelike, animated characters. Although we feel that there is no real value in this approach we nevertheless decided to implement simple visual Pedagogical HAPA agent and let students decide if they will activate it or not, using the on/off switch button. But, although maybe "fun" to look at in the beginning, over the time the visual character stands in the way



```
for (int i = ?; i < 10; i++){}
```
What should be the starting index?
Hint: the first element of the Fibonacci sequence is often denoted as f0

Figure 2: Visual representation of a Pedagogical HAPA agent.

of solving the assignment. This was somehow confirmed during the experimental phase with students. They distract the student from concentrating on the problem in question, and in the extreme case, may negatively affect his/her willingness to use the system.

Both pedagogical agents are capable of adapting to each individual student. Agents track a set of information about the student, including his/her personal data (such as class and age), the ratio of correct and incorrect solutions to each code completion problem, and the student's grade for each lecture topic. Based on the accumulated data agents can intervene if the student's success rate becomes unsatisfactory. For example, if the student gives to many wrong answers to questions regarding *for loops*, the pedagogical agent will recommend additional learning material. Additionally, it will repeat the appropriate code completion tasks until a certain success threshold is reached.

We believe that if students do not know if agent gives correct or wrong directions and hints, it will additionally motivate students to critically think and assess their knowledge. Also in future real working environments they will face different helpful but also malicious colleagues who will maybe suggest them wrong procedures and steps. So we would like to put students in unexpected situations and motivate them to reassess their knowledge and skills.

4 HAPA implementation and evaluation

Previously described functionalities of HAPA were used to guide the implementation process. For example, harvesting is a process that can and should be distributed and executed in parallel. Then, students should be able to interact with and use HAPA through a web interface. And, like all web-based systems, HAPA should be resilient to hardware and software failures, malicious attacks, etc. Given these implementation requirements, and its popularity in developing software agents and multi-agent systems, Java has been chosen as the implementation platform for HAPA.

4.1 Helpful and misleading hints

In order to provide the reader with a better insight into the evaluation of HAPA, some examples of the prepared code completion tasks are presented here. The two given tasks are tailored to topics on *for loops* and *classes* in Java, respectively. Helpful and misleading hints assigned to each task are also presented and discussed.

The task tailored to the topic on *for loops* in Java requires the student to complete a program for calculating the first 10 members of the *Fibonacci sequence*. The skeleton program presented to students is shown in Listing 1.

Listing 1: Code completion task related to *for loops*

```
class Fig {
  public static void main(String[] args) {
    int[] f = new int[10];
    // TODO : implement the for loop here
    print(f); } }
```

Based on this skeleton, the following set of helpful and misleading hints for pedagogical agents have been prepared [8].

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."
4. *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."
5. *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?"
6. *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."

By suggesting that f_0 is the first element of the Fibonacci sequence in hint 4, the misleading agent tries to suggest the improper usage of 0 for the initial value of *i*. In the expression $f_i = f_{i-1} + f_{i-2}$, this decision would cause the index to go out of the array bounds. Similarly, in hint 5, the agent suggests that the student should use 10 as the final value of *i* (note the expression *i ≤ ?*), disregarding the fact that Java array indexes are 0-based. The final hint 6 is there to confuse the student, since both *++i* and *i++* are correct.

Listing 2 shows the skeleton for a more complex task tailored to classes, fields and methods. The given class represents a rectangle, defined by its upper-left point *(x, y)*, *width*, and *height*. The student's task is to write a method that calculates the rectangle area. The focus is on the proper definition of the method's input parameter and the return value.

Listing 2: Code completion task related to classes.

```
class Rect {
  private float x, y, width, height;
  public Rect(float x, float y, float width, float height) {
    // the parameters are saved into corresponding fields
  }
  // TODO : calculate the rectangle area here
}
```

For this example the following set of hints are defined. Hints 7 and 8 are used by the helpful, while hints 9 and 10 are used by the misleading agent.

7. "To calculate the area, you need width and height, both of which are available as fields. Remember that a method can access its object's fields without limitations."
8. "While writing the method, remember that somebody needs to use the result; outputting it on the screen won't be of much use to anybody!"
9. "To calculate the area, you need both width and height. Make sure your method has access to these values!"
10. "Remember that long command for outputting values on screen? Here's a hint: it starts with 'System'."

Given misleading hints are based on our long-term experience in conducting exercises for the introductory Java programming course. It has been observed that initially, a relatively large portion of students has the problem of grasping the concept of fields, methods, and method arguments. That is, they tend to specify fields as method arguments, rather than to use them directly. This is what the misleading hint 9 tries to suggest. The given constructor implementation that receives both width and height also works in favor to this suggestion.

Similarly, when asked to calculate some value (in this case, the rectangle area), beginner students often tend to just print the value on screen, rather than to return it from the method. This is what hint 10 tries to lead them to.

After preparing these and many other code completion tasks and hints, the system has been evaluated in practice. We expected that evaluation results would obtain adequate feedback necessary to continue our efforts and improve system's functionalities.

4.2 Evaluation results

Following the implementation of HAPA, an evaluation of the system was conducted during school year 2013/14. The main goal of the evaluation was to examine the effects of helpful and, more importantly, misleading pedagogical agents.

Because the current implementation of HAPA is the first prototype of the system, the evaluation was performed on a small group of self-motivated students. 24 second-year students were selected, on the basis of their previous programming experience, as well as their scores in previous programming-related course at our Department of mathematics and informatics, Faculty of Sciences (DMI) and 18 students from Department of information technology at the Higher School of Professional Business Studies (DMT).

During the semester, from time to time the students were given a number of Java code completion tasks to solve. Once it presents a task, the system waits for the student's input for a certain amount of time. If no input is detected, a pedagogical agent provides a hint for solving the task. As noted earlier, helpful and misleading agents take turns at random time intervals, and the student is never sure with which agent he/she is interacting. Moreover, students were not even aware of the fact that there are two types of pedagogical agents.

Students' responses and actions were logged by the system. After the testing an analysis of the log was conducted in order to determine whether the hints have had any influence onto the students' though processes. As they used the system partially in blended learning style, for some lessons to obtain their opinion we did not use classical form of questionnaire. We just interview them and ask them about their opinion. We expected that in such friendly atmosphere they would be completely honest. The summary results are presented below.

Students from DMI showed better results and are were more eager in using HAPA system. Students from DMT made a lot of mistakes and were in majority of cases frustrated by using the system. More detailed explanations are given in the rest of the section.

Group DMI_G1 consists of eighteen students from DMI, they have had previous programming experience, have achieved great scores in a previous programming course and were among the best ones. Group DMI_G2 consists of six other students that have previous programming experiences but did not pass previous programming course.

12 students from DMI_G1 were very cautious and thought critically most of the time, and recognized and ignored hints from misleading agent. Other 6 students also employed critical thinking and in majority of cases recognized misleading hints. But in several cases when they were no sure about the proper choice they accepted hints form misleading agent.

4 students from DMI_G2 blindly followed all hints and did not think whether they are correct or not. 2 other students were a little bit confused, they thought that some hints were wrong, but as they were not absolutely sure about that they decided to follow all hints. This is probably the result of their inappropriate knowledge.

In the next several cases we will illustrate some specific students' behaviors. One student among the best students had no problem in solving the tasks, and, according to his own account, rarely considered hints. This is because he had already learned Java programming in high-school, and so could understand and solve the problems without any assistance.

Another student has considered the hints, but was also able to solve majority of the tasks without any assistance. However, he indicated that hint 7 helped him to solve the problem. He has also correctly observed that some of the presented hints were wrong.

A more interesting situation was with the two students who had no previous Java programming experience and have low scores on previous programming course. One of them took hint 4 for granted and received the *array-index-out-of-bounds* exception. On the second attempt, she was given hint 1 and was able to complete the task successfully. A similar scenario has been observed with the fourth student. By accepting hint 9, he initially wrote a method that accepts both width and height as input parameters. Then, after receiving hint 7, he was able to complete the task successfully.

The system was also tested and used among 18 students of the DMT. Involved students were programming beginners and they used system after completion of the first programming course.

The results of the experiments showed that the students of DMT had more trust in agents, and made more mistakes based on the hints of misleading pedagogical agents. Majority of them (even 13) blindly followed suggestions and believed that hints were well-intentioned. Other were more or less confused and do not know if can trust or not to obtained hints.

Several interesting conclusions can be drawn from these results. If students are unable to solve the problem on the first try, they have the tendency to trust the agent's/tutor's hints. This is because they have never encountered a misleading agent/tutor before and do not anticipate such behavior. The student who had noticed the wrong hint was confused but assumed that it was an implementation error. For some future work, it could be beneficial if students were informed that some hints might be intentionally misleading.

Secondly, students who are not confident in their Java skills find the presence of a virtual agent/tutor "reassuring." Additionally, the misleading behavior and the attempt of the agent to "trick them" transform HAPA into a kind of a game with the goal of beating the system. Both of these effects have a significant positive impact onto the students' motivation to use the system and employ critical thinking. These are the exact goals set for our proposed system.

Finally, results suggest that students from DMT were more misled by pedagogical agents than students from DMI. A reasonable explanation could be that students from DMI are better (achieved better results in secondary school, passed with high marks first programming course) and they are generally highly motivated (usually students from DMI are more ambitious about their future career and jobs) to master their programming knowledge and skills.

Both groups of students have the similar attitude to visual representation of pedagogical agent(s) and their use in the system. As we mentioned, students were allowed to explicitly (de)activate visual forms of agents. In the beginning of using system it was attractive to them to see agents and communicate with them so they intensively used them. Lately, especially when they did not know how to solve task and as they were getting more tired and nervous about hints, visual agents irritated them and they decided to switch off visual forms of agents.

5 Conclusions and future work

The agent technology has been recognized as a useful tool in a wide variety of domains. For e-learning and tutoring systems, harvester and pedagogical agents are of the special interest.

The main contribution of this paper is the proposal of a new e-learning system named HAPA that incorporates both harvester and pedagogical agents. Harvester agents, along with the Classifier module, are designed to collect the best-suited Java source code examples from the web, and tailor them to particular lessons within a course.

A more important functionality, however, is achieved by defining two new sub-types of pedagogical agents – helpful and misleading. As noted, the helpful pedagogical agent provides correct suggestions and hints for the problem in question. On the other hand, the misleading agent tries to guide the problem solving process in a wrong direction, by offering false suggestions and hints. The main aim for this approach is to motivate students not to follow the agent's directions blindly, but instead to analyze both the problem and the suggestions thoroughly and employ critical thinking. According to our knowledge, none of the existing e-learning systems use this kind of helpful and misleading pedagogical agents in combination with harvester agents.

HAPA is currently realized as a prototype. Future improvements will be concentrated on integrating it into our existing, fully-featured web-based e-learning architecture Protus [10, 19], but it can also be integrated in some other types of available learning systems [14, 18]. In order to achieve adaptability and personalization, Protus incorporates several models, including: *Domain model*, which serves as a storage of the learning material, *Student model* for maintaining both static and dynamic information about each individual student, *Application model*, which applies different strategies on the input received from the learner model in order to ensure efficient personalization, and the *Adaptation model* that follows the instructional directions provided by the application module in order to organize learning resources into a navigational sequence tailored to the particular learner.

Obviously, modules that comprise HAPA fit nicely into the organizational models of Protus. Harvester agents, along with the Classifier module, can be used to obtain and generate learning material for the domain model. The learning material stored in the domain model consists of individual lessons, which are further decomposed into tutorials, accompanying examples, and tests. Therefore, harvester agents and the Classifier module can be used to collect examples and tests. Proposed helpful and misleading pedagogical agents can be integrated into the adaptation model. This integration will harness the benefits of both architectures. The resulting system will be capable of providing high-quality tasks and examples, and exhibiting adaptive and personalized behavior. It will offer motivational pedagogical agents that guide students through the learning process and encourage critical thinking.

Having in mind that usual way of employing pedagogical agents is to help students to systematically test their self-confidence and knowledge and somehow keep them less active and expecting constant positive help we decided to apply the opposite way. We believe that if students do not know if agent gives correct or wrong directions and hints, will additionally motivate students to critically think. Also in future real working environments they will face different helpful but also malicious colleagues who will maybe suggest them wrong procedures and steps.

So we would like to put students in unexpected situations and prepare them to face rather stress and competitive real working environments.

Acknowledgment

This work was 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."

Bibliography

[1] Barcelos, C.F., and Gluz, J.C.(2011); An agent-based federated learning object search service. *Interdisciplinary journal of e-learning and learning objects 7*, 37-54.

[2] Capuano, N., Marsella, M., Salemo, S. (2000); ABITS: an agent based intelligent tutoring system for distance learning. In *Proceedings of the International Workshop in adaptative and intelligent web-based educational systems*, 17-28.

[3] Cardoso, J., Guilherme, B., Frigo, L., and Pozzebon, L.B. (2004); MathTutor: a multi-agent intelligent tutoring system. In *First IFIP conference on AIAI* , 231-242.

[4] Gago, I.S., Werneck, V. M., and Costa, R. M. (2009); Modeling an educational multi-agent system in MaSE, In *Proceedings of the 5th international conference on active media technology (AMT'09)* , 335-346.

[5] Haake, M., and Gulz, A. (2008); Visual stereotypes and virtual pedagogical agents. *Educational Technology & Society 11*, 4: 1-15.

[6] Heidig, S., and Clarebout, G. (2011); Do pedagogial agents make a difference to student motivation and learning? *Educational Research Review 6*, 27-54.

[7] Heller, B., and Procter, M. (2010); Animated pedagogical agents and immersive worlds: two worlds colliding. *Emerging Technologies in Distance Education*, 301-316.

[8] Ivanović, M., Mitrović, D., Budimac, Z., Vesin, B., and Jerinić, L. (2014); Different roles of agents in personalized learning environments. In *10th International Conference on Web-Based Learning (ICWL 2011), LNCS*, Springer, 7697: 161-170.

[9] Ivanović, M., Mitrović, D., Budimac, Z., and Vidaković, M. 92001); Metadata harvesting learning resources – an agent-oriented approach. In *Proceedings of the 15th International Conference on System Theory, Control and Computing (ICSTCC 2011)*, October 2011, 306-311.

[10] Klašnja-Milićević, A., Vesin, B., Ivanović, M., and Budimac, Z. (2011); Integration of recommendations and adaptive hypermedia into Java tutoring system. *Computer Science and Information Systems 8*, 1: 211-224.

[11] Kobayashi, M., and Takeda, K. (2000); Information retrieval on the web, *ACM Computing Surveys 32*, 2: 144-173.

[12] Linstead, E., Bajracharya, S., Ngo, T., Rigor, P., Lopes, C., and Baldi, P. (2009); Sourcerer: mining and searching internet-scale software repositories. *Data Mining and Knowledge Discovery 18*, 2: 300-336.

[13] Mitrovic, A., and Suraweera, P. (2000); Evaluating an animated pedagogical agent. In *Intelligent Tutoring Systems*, Springer, 73-82.

[14] Ocepek, U., Bosnic, Z., Serbec, I.N., and Rugelj, J. (2013); Exploring the relation between learning style models and preferred multimedia types. *Computers & Education 69* , 343-355.

[15] Orzechowski, T. (2007); The use of multi-agents' systems in e-learning platforms. In *Siberian conference on control and communications (SIBCON'07)*, 64-71.

[16] Prieta, F. D.L., and Gil, A.B. (2010); A multi-agent system that searches for learning objects in heterogeneous repositories. In *Trends in Practical Applications of Agents and Multiagent Systems, Advances in Intelligent and Soft Computing*, Springer, 71: 355-362.

[17] Sharma, S., Gupta, J.P.(2010); A novel architecture of agent based crawling for OAI resources. *International Journal of Computer Science and Engineering 2*, 4: 1190-1195.

[18] Stuikys, V., Burbaite, R., and Damasevicius, R. (2013); Teaching of computer science topics using meta-programming-based GLOs and LEGO robots. *Informatics in Education 12*, 1: 125-142.

[19] Vesin, B., Ivanović, M., Klašnja-Milićević, A., and Budimac, Z. (2013); Ontology-based architecture with recommendation strategy in java tutoring system. *Computer Science and Information Systems 10*, 1: 273-261.