Vojnotehnicki glasnik/Military Technical Courier

ISSN: 0042-8469

vojnotehnicki.glasnik@mod.gov.rs

University of Defence

Serbia

Šimi P., Goran

JAVA PROBLEM-BASED LEARNING

Vojnotehnicki glasnik/Military Technical Courier, vol. 60, núm. 1, enero-marzo, 2012, pp. 57-69

University of Defence

JAVA PROBLEM-BASED LEARNING

Šimić P. *Goran*,
University of Defence, Military academy, Center for Simulations and Distance Learning, Belgrade

FIELD: Computer Sciences, E-learning & ODL (Opet & Distance Learning) Technologies

ARTICLE TYPE: Original Scientific Paper

Sažetak:

The paper describes the self-directed problem-based learning system (PBL) named Java PBL. The expert module is the kernel of Java PBL. It involves a specific domain model, a problem generator and a solution generator. The overall system architecture is represented in the paper. Java PBL can act as the stand-alone system, but it is also designed to provide support to learning management systems (LMSs). This is provided by a modular design of the system. An LMS can offer the declarative knowledge only. Java PBL offers the procedural knowledge and the progress of the learner programming skills. The free navigation, unlimited numbers of problems and recommendations represent the main pedagogical strategies and tactics implemented into the system.

Key words: *problem based learning, e – learning, intelligent tutoring systems*.

Introduction

This paper describes the case study in self-directed problem-based learning. Problem-based learning (PBL) is a student-centered instructional strategy in which students (individually, or organized in groups) solve problems and reflect on their experiences. PBL is strongly founded in a specific domain of expertise and it emphasizes critical thinking and problem-solving skills of students.

Different domains need different approaches in the design of PBL systems. The common characteristic of these systems is an assigned problem and learners have to have some procedural knowledge needed to solve this problem. In a PBL scenario, the PBL system plays the teacher's role and it already has to have this knowledge. This is the start point of the presented research.

The next section briefly overviews related work of other authors relevant to the design and development of the PBL system, providing the context within which the presented studies are best understood. Section

goran.simic@mod.gov.rs

3 presents the motivation. The system design and architecture are analyzed in Section 4. The case study is described in Section 5. The system evaluation is presented in Section 6.

E-Learning and PBL

Intelligent tutoring systems (ITSs) in different domains represent the applications of computer-supported PBL. There are many PBL ITSs in medicine: Slide Tutor [1] is designed for learning dermatopathology. A student makes his/her own hypothesis about the assigned problem. He/she chooses the most appropriate concepts from the domain ontology and tries to prove his/her hypothesis. CIRCSIM-Tutor [2] teaches cardiovascular physiology by a Socratic-style dialog with the student to help him/her in reasoning towards the correct solution, and many others. Cognitive Tutor [3] is a well-known mathematic tutor which builds a cognitive model of a student as he/she interacts with the application. This ITS profiles the student and it performs system adaptation according to student's solutions of given problems. In a KERMIT [4], PBL and student modeling are implemented (Knowledge-based Entity Relationship Modeling Intelligent Tutor) by constraints. Some PBL systems are accessible over the Internet. For example, ELM ART [5] is a Web-based tutor designed for tutoring in LISP programming language. The ELM ART problem adaptation is based on an episodic learner model. An episode represents the parts of the learner session related to the concrete problem and to the earner's attempts to solve it.

The adaptation of the PBL process in the ITS is based on the specialized learner model. The adaptation means the problem, which the ITS assigns to the learner, is changeable according to the actual state of the learner's model. This model will be changed every time when the ITS processes the leaner's solution.

Learning management systems (LMSs) represent the most widely used e-learning systems over the Web. They improve both the self-paced and the instructor-led learning processes [6]. Unfortunately, these systems do not support PBL. LMSs are designed to be domain-independent systems, with the rich palette of administration functionalities. They are focused on delivering reusable, well-structured learning content. Therefore, just the declarative knowledge of the individual learner can be evaluated in an LMS. Fortunately, LMSs are well-standardized systems (mainly according to SCORM 2004 standard). They have good cooperative possibilities. The new functionalities can be added as new modules of the LMS, or as separate applications which interchange the data and other resources with the LMS.

Description of the problem

The aim of the research described in this paper is to develop the separate application which supports LMSs for PBL. Based on real needs, one implementation is considered: the course about Java programming.

Java is one of the most popular programming languages, but there are just few systems for Java tutoring on the Web. Some of them are focused on learning specific skills. For example, Swing Tutor [7] is designed for learning how to make interfaces by using Java GUI classes. There are some disadvantages of using these systems. The learners cannot navigate through the system. They are leaded step by step through the learning space without the possibility of navigating back. Neither the system shows the "whole Figure" of the course (structure, organizations, tasks and goals), nor the students can see their results from previous attempts. The non-intuitive and complex user interface represents additional disadvantages of these two tutors.

Other example of Java tutoring is Java Sprint [8] - the commercial pluggable component for Eclipse IDE. Differently from other Java ITSs, Java Sprint covers the most important Java programming concepts. Unfortunately, this tool cannot be used over the Web, and it is designed just for one client per application. The dependence of the Eclipse platform represents another disadvantage of this tool.

The facts mentioned above motivate us to develop the custom components for PBL of Java programming. The domain-based system design and usage are represented in this paper.

Environment and the PBL system

The general architecture of the PBL system represents the start point of research. Every PBL system consists at least of five modules (minimal number of modules). The learners interact with the system through the *communication* module. This is the most frequently mentioned module in the architecture of e-learning systems [9-12]. The *problem generator* and the problem solver represent the other necessary modules in PBL systems. The first one generates the problem for the learner. The *solution generator* generates the system solution based on a delivered problem. The *expert module* checks the learner solution by comparing it with the system solution. The differences are used for diagnosing and for updating the state of the *learner model*. This is the responsibility of the expert module. In the next iteration, the problem generator modifies the problem based on the changes in the learner model. This means the learner model makes the system adaptable according to the learner's actual needs.

Extended model

Note that the basic architecture does not contain any explicit mechanism which can provide help to the learner. Short feedback messages or other similar mechanisms are used in this case. If the PBL system is a part of some other system (e.g. LMS), the environmental system learning resources can be used for this kind of support. In this case, the basic architecture has to be extended with new concepts (Fig. 1).

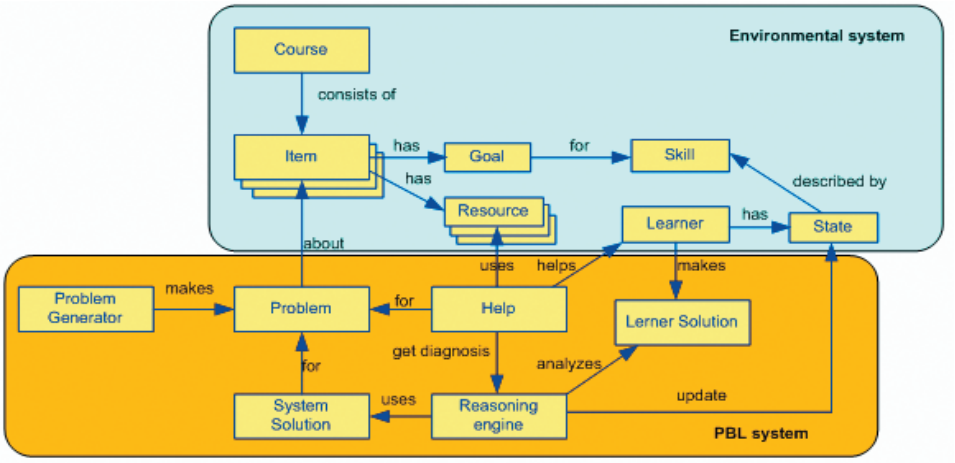

Figure 1 – The additional concepts which improve contextual help

These concepts are designed according to the actual e-learning standards implemented in the most of LMS [10,11]. The item represents the organizational unit of the course (learning unit). The item has a goal which is to master a specific skill. The learner's state is also described by his skills.

If the problem is related to the course item, the variety of problem types will be proportional to the course fragmentation. More items need more problem types. On the other side, the help system will be more effective if the item resources describe smaller pieces of knowledge. In this way, help will be more contextual.

When the learner uses the PBL system, his state permanently changes. The system helps the learner with a specified type of the problem by using the resources related to the same item as the problem. This help is also harmonized with the learner's state. Since the state is described by skill levels, the system tries to find the most appropriate resources to improve them. Also the system tries to get a diagnosis for a concrete problem. The reasoning engine is used for performing these advanced tasks.

PBL in Java Programming

The design of the Java Programming PBL system is based on the language building blocks. The class (*UDClass*) represents the main language concept (Fig. 2). It consists of data members (attributes) and function members (methods). The research is focused on the basic programming skills. This means that the learners are able to define the variables and methods and to use them.

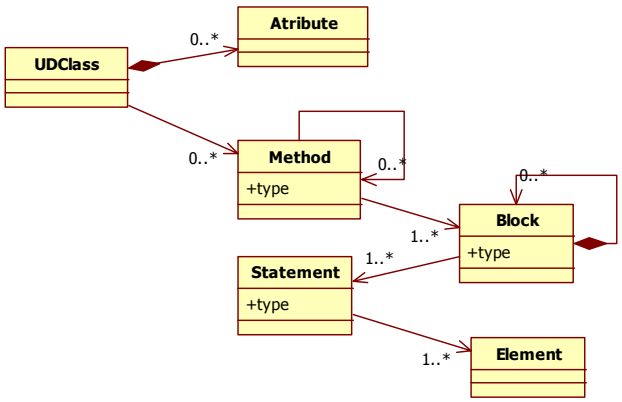

Figure 2 – Used concepts in Java Programming PBL

The method contains one or more blocks. There are different types of methods, based on visibility, number of arguments, return value and method content. They can consist of functional calls – other methods, defined in the same class or in other classes. Methods consist of blocks. The four block types are considered: sequential, conditional, cyclic and single statement blocks. The blocks can also contain embedded blocks. The blocks consist of statements. There are also different types of statements: declarations, definitions, value assignments and examinations, etc. Each statement is made of elements (e.g. variable types, names, values and different kind of operators). All concepts mentioned above are encapsulated as classes in the *Java PBL*.

Problem types

Different problem types are defined regarding the concepts mentioned above. Problems can also be layered regarding their complexity. The next illustration (Fig. 3) demonstrates the relations between the problem types and the programming concepts. The basic level problems are related to the basic statements such as variable or array definitions. The value assignments and data retrieving are also used for the basic level problems.

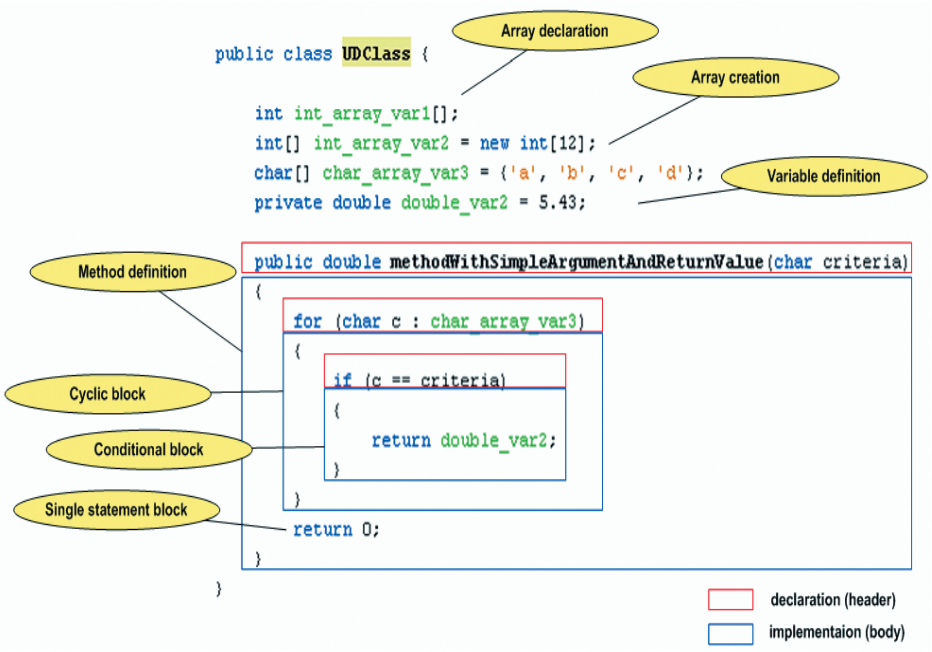

Figure 3 – Sample class concepts used in considerations

The 2nd level problem types include the implementation of different blocks (sequential, cyclic, etc.). The whole method implementations are included in the 3rd level problems. There are several problem types regarding the method properties (e.g. type and number of arguments, required functionality and return values). The highest level problems require the definition of a whole class (or classes) to accomplish more general functionalities.

The class content is hierarchically structured. Every block and every method generally consists of a declaration and an implementation part (named header and body). This characteristic is very important for generating the system solution and for analyzing the learner solution. Note that the basic architecture does not contain any explicit mechanism which can provide help to the learner. The short feedback messages or other similar mechanisms are used in this case. If the PBL system is a part of some other system (e.g. LMS), the environmental system learning resources can be used for this kind of support. In this case, the basic architecture has to be extended with new concepts.

The system generates the problem dynamically. The names, types and values (of the variables, methods and classes) are defined randomly. These data represent the problem parameters. The problem consists of three or more parameters. The system uses these data to make the system solution and to check the learner solution.

For example, one of the 3rd level problem types is the single argument method with the IF clause. This kind of problems consists of 8 parameters: the method return type and name, the argument type and name, the criteria value, logical operator type (used in the IF clause), and two return values (when the IF clause is true and when the IF clause is false). The learner solution diagnostic in the PBL system is based on checking the presence of these parameters.

Processing of the learner solution

The PBL system processes the learner solution in several stages (Fig. 4). The solution embedding and compiling are the first two stages of the solution processing.

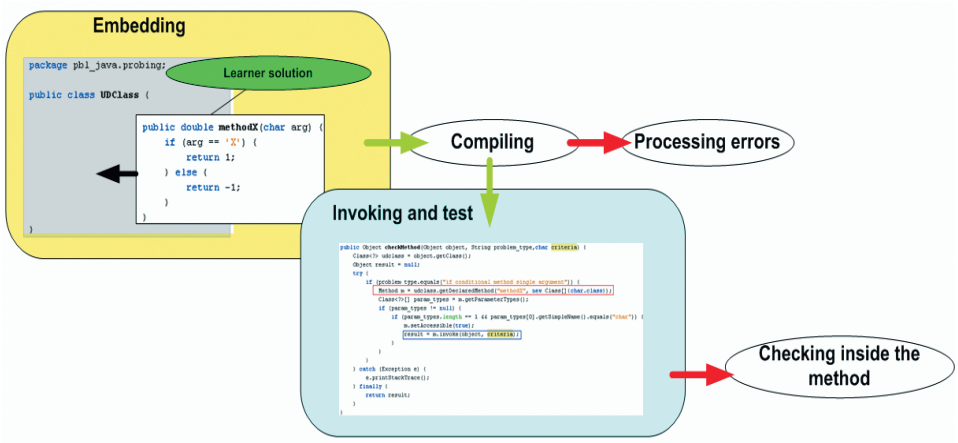

Figure 4 – Processing of the learner solution

Embedding means that the system completes the learner solution with the necessary code in order to compile it. The solutions of the 1st level problems (see the previous paragraph) have to be embedded in the empty class declaration. The 2nd level problem solutions require double embedding – by method and (method) by class. The 3rd level solutions have to be wrapped by class.

The system checks the solution syntax by the compiler. If the compiling results are without any exception, then the solution syntax is correct. On the other hand, the PBL system processes the errors by catching the diagnostics (*javax.tools.Diagnostic*) returned from the compiler, and customizes these to the learner.

If the solution syntax is correct, the PBL system performs the next phase – invoking and test. Java reflection classes are used for this purpose (*java.lang.reflect* package). These classes provide the instantiating and inspecting of the user-defined classes. The system is able to check declarations (types and names of class members) as well as the method behavior.

The 2nd and higher type solutions are tested by invoking the requested methods. Then, the PBL system passes different parameter values to the method, picks the results, and compares these with the expected values. If the compared values are different, the system performs the highest level checking – semantical checking of the code inside the method. In this case, the method body is parsed (as previously described) and processed by an appropriate expert class (Fig. 5).

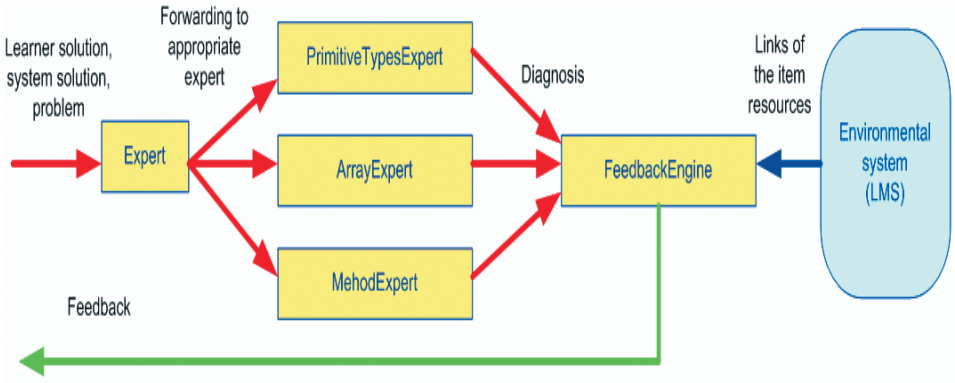

Figure 5 – Processing of the learner solution

The expert plays the role of a dispatcher by recognizing the problem type and forwarding the input data to the specialized expert. It compares the solutions and makes diagnosis. The feedback engine requests the links of the item resources based on the problem and diagnosis. Finally, the PBL system delivers the feedback messages, the helpful links and the system solution to the learner.

Example of usage

The complex PBL system structure does not influence the user interface simplicity. *Java PBL* could be used as a desktop, or Web (JSP) application. Both types use the same PBL engine. The user interface communicates with the engine through the engine API (implemented as a separate class – UML design pattern *Façade*). The learner has free navigation through the PBL resources. Different problem types are represented to him as hyperlinks to the specified problem solving content (Fig. 6).

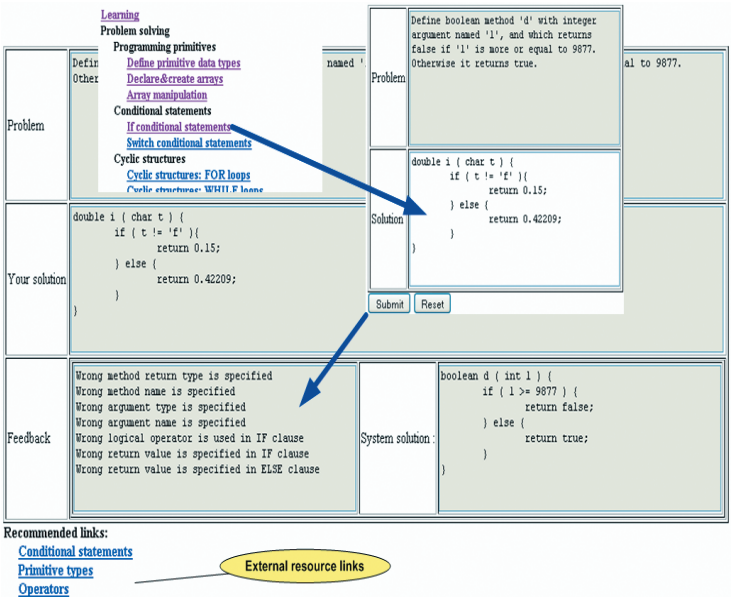

Figure 6 – Usage of *Java PBL*

Based on the user choice (learning topic), *Java PBL* presents a dynamically generated problem to the learner. If the learner tries to reload the page, or return to the previous window, the new problem will be generated. Thus, the learner is pedagogically directed to use the system in an appropriate way.

The learner tries to solve the problem. After the user submits his solution, the system processes it as described above. In the given example, the learner's method does not return expected results, and the PBL system performs the highest level of checking. Therefore, the feedback contains a detailed description of the semantical errors. The errors are formalized in the system. As the problems, they are also related to the learning resources. The system uses this characteristic to show the learner the most appropriate links in the form of recommendations. The learner is free to choose whatever he wants. He can continue with problem solving, or he can study the recommended materials. The system registers all of the learner's actions and these data can be used for different kinds of later reporting.

Conclusions

The presented study proves that the PBL systems can be a complement of the LMS. The actual LMS does not have support for PBL. Owing to the modular architecture and standardization of the learning resources, these two kinds of systems can coexist. The LMS learning resources are designed to provide learners with the declarative knowledge (definitions, explanations, illu-

strations, lectures). But this kind of knowledge (*knowing-what*) is not helpful enough in solving concrete domain problems (*knowing-how*). On the other side, if the learner does not have enough declarative (descriptive) knowledge, he will not be able to perform tasks related to this knowledge. The different approaches can be applied in using these systems. In hereby studies, the LMS learning resources are used for improving PBL. The PBL system can also be considered as a separate component of the LMS system. The irrefutable fact is that the learning process is enriched by using both of them.

Java PBL provides free navigation to the learner. He can change the problem type whenever he wants. Irrespective of the navigation possibilities, the learner can self pace the learning progress. The applied pedagogy strongly depends on the nature of the domain. The learner progress in programming is performed by a number of attempts. This domain needs much more experience than procedural knowledge.

The open architecture of the presented PBL solution provides the possibility of adding new modules. In this way, new problem types can be separately developed and easily plugged into the system and linked with LMS materials.

Reference

[1] Crowley, R., and Medvedeva, O., *An Intelligent Tutoring System for Visual Classification Problem Solving*. *Artificial Intelligence in Medicine*, Elsevier, 2006, 36 (1), 85-117

[2] Mills, B., Even, M., and Freedman, R., *Implementing directed lines of reasoning in an intelligent tutoring system using the atlas planning environment*. *International Conference on Information Technology*: ITCC 2004, 729-733

[3] Anthony, Lisa; Yang, Jie; Koedinger, Kenneth R., *Evaluation of Multimodal Input for Entering Mathematical Equations on the Computer, ACM Conference on Human Factors in Computing Systems* (CHI 2005), Portland, OR, 4 Apr 2005, 1184-1187

[4] Suraweera, P., Mitrovic, A., *An Intelligent Tutoring System for Entity Relationship Modelling*, *International Journal of Artificial Intelligence in Education* (*IJAIED*), 2004, 14, 375-417

[5] Weber, G., Brusilovsky P., *ELM-ART: An Adaptive Versatile System for Web-Based Instruction, International Journal of Artificial Intelligence and Education*, 2001, Vol.12, 351-384

[6] Beck, J., Stern, M., Haugsjaa, E., *Applications of AI in Education*, *ACM Crossroads*, 1996, Vol. 3, No. 1, 11-15.

[7] Java tutoring systems: Swing Tutor, available at: http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/SwingTutorLinksV2.html

[8] Java pluggable component for Eclipse IDE, available at: http://www.javasprint.com/

[9] Brusilovsky, P., *A Distributed Architecture for Adaptive and Intelligent Learning Management Systems*, in *Proceedings of AIED 2003 Workshop: Towards Intelligent Learning Management Systems*, Sydney, 5-13.

[10] Šimić, G., *Modelovanje korisnika u sistemima za upravljanje sadržajem*, Vojnotehnički glasnik/Military Technical Courier, Vol. 53, No. 1, pp. 59-68, Ministarstvo odbrane Republike Srbije, Beograd 2005.,

[11] Shimic, G., and Jevremovic A., *Problem-based learning in formal and informal learning environments, Interactive Learning Environments*, Taylor & Francis, 2010,

[12] http://www.tandfonline.com/doi/abs/10.1080/10494820.2010.486685#preview

UČENJE PROGRAMSKOG JEZIKA JAVA REŠAVANJEM PROBLEMA

OBLAST: računarske nauke, e-learning & ODL (Open & Distance Learning) tehnologije

VRSTA ČLANKA: originalni naučni rad

Sažetak:

Ovaj rad opisuje sistem za individualno učenje rešavanjem problema čiji je naziv Java PBL. Jezgro Java PBL je ekspertski modul. Ovaj modul uključuje specifičan domenski model, generator problema i generator rešenja. U materijalu je predstavljena celokupna sistemska arhitektura. Java PBL može da funkcioniše kao samostalna aplikacija, ali je dizajnirana i da pruži podršku sistemima za upravljanje učenjem (LMS). Modularni dizajn sistema obezbeđuje ovu funkcionalnost. Sistemi za upravljanje učenjem mogu da ponude korisnicima samo deklarativna znanja. Java PBL nudi proceduralna znanja i napredovanje veštine programiranja kod korisnika. Slobodna navigacija, neograničeni broj zadataka (problema za rešavanje),preporuke predstavljaju glavne pedagoške strategije i taktike primenjene u sistemu.

Uvod

Učenje rešavanjem problema (PBL) predstavlja strategiju u nastavi usmerenu direktno na studenta, u kojoj on rešava problem (zadatke) i kroz to reflektuje svoje iskustvo (znanje). PBL je zasnovano na domenskoj ekspertizi, izgrađuje kritičko razmišljanje i veštinu rešavanja problema kod studenata. Da bi student rešio problem, on mora da poseduje odgovarajuće proceduralno znanje. U učenju rešavanjem problema PBL aplikacija ima ulogu nastavnika, tako da i u njoj mora da bude ugrađeno isto to znanje.

Elektronsko učenje i učenje rešavanjem problema

Učenje rešavanjem problema implementirano je u inteligentnim tutorskim sistemima (ITS). U praksi preovladavaju ITS iz domena medicinskih nauka: Slide Tutor [1] namenjen za učenje dermatopatologije. CIRCSIM-Tutor [2] namenjen je za učenje kardiovaskularne psihologije. Pored medicinskih, česti su matematički ITS, na primer, Cognitive Tutor [3]. KERMIT [4] je informatički tutor za učenje pravljenja modela objekti – veze. Neki od ITS dostupni su preko interneta, na primer, ELM ART [5] je ITS namenjen za učenje LISP programskog jezika.

Sistemi za upravljanje učenjem (LMS) predstavljaju preovlađujuće sisteme za e-učenje na Webu. LMS omogućavaju i individualno i učenje

vođeno od strane nastavnika [6]. Ali LMS nemaju mogućnost za učenje rešavanjem problema. LMS su standardizovani (uglavnom sa SCORM 2004 standardom), tako da su otvoreni za saradnju s drugim sistemima. Nove funkcionalnosti mogu da se dodaju, bilo kao novi sistemski moduli ili kao posebne aplikacije koje sa LMS razmenjuju podatke studenata i druge resurse.

Opis problema

Fokus istraživanja opisan u materijalu je razvijanje posebne PBL aplikacije koja će predstavljati podršku za LMS. Zbog realnih potreba, izabran je kurs iz Java programskog jezika.

Iako je Java jedan od najpopularnijih programskih jezika, postoji samo nekoliko sistema za učenje Java na Webu. Ovi sistemi fokusirani su samo ne specifične delove jezika. Na primer, Swing Tutor [7] usmeren je isključivo na izgradnju korisničkih interfejsa. Ovaj sistem ne dozvoljava slobodnu navigaciju u prostoru učenja niti studenti vide prethodno ostvarene rezultate. Drugi primer je dodatni modul za Eclipse razvojno okruženje. Ovaj modul pokriva tutorstvo za najvažnije koncepte Java. Nažalost, nije dostupan posredstvom Weba. Zavisnost od Eclipse okruženja drugi je nedostatak sistema. Oba sistema su izolovani od drugih okruženja za e-učenje. Iznete činjenice bile su motiv za razvoj vlastitih komponenti za PBL Java programskog jezika.

Okruženje i sistem za učenje rešavanjem problema

PBL sistem, pored komunikacionog modula, sadrži generator problema i rešenja, ekspertski modul za kontrolu rešenja i korisnički model u koji se ažurira stanje (rezultati) korisnika. Osnovni mehanizam tutorstva su povratne poruke o sintaksičkim i semantičkim greškama i prikaz tačnog rešenja.

Prošireni model sistema

Ako je PBL sistem okružen na primer LMS-om, resursi učenja u LMS mogu da postanu deljivi. Pošto su LMS usklađeni sa SCORM 2004 standardom, osnovna komponenta item, koja može da predstavlja tematsku celinu, lekciju, povezuje se sa tipom problema. Ovaj princip korišćen je za deo PBL sistema za preporučivanje sadržaja učenja.

Učenje rešavanjem problema u Java programiranju

Java PBL namenjen je za izučavanje osnovnih koncepata programiranja u Javi, tako da je definisana jedna klasa (UDClass) kroz koju se proveravaju korisnička rešenja. U sistemu su kroz klase implementirani koncepti atributa i metoda, dalje blokova naredbi, koji imaju svoje izvedene klase prema vrsti – ciklične strukture, kontrole toka, ili proste sekvencijalne strukture i na kraju – same naredbe i njihovi elementi.

Vrste problema

Zahvaljujući strukturiranosti modela, u Java PBL definisana su 4 težinska nivoa problema. Prvi nivo je definisanje varijabli različitih tipova. Drugi nivo je da se implementiraju različite strukture (razgranate ili ciklične), treći nivo je definisanje metoda klase, koje kombinuju sve prethodno. Najviši nivo je da se definiše čitava klasa, koja sadrži različite metode.

Obrada korisničkog rešenja

Obrada korisničkog rešenja zavisi od nivoa problema. Sintaksička provera rešenja vrši se propuštanjem izvornog koda rešenja kroz Java kompajler. Za jednostavnije tipove problema rešenje mora da se ugnezdi u odgovarajuću strukturu kako bi se kompajliralo. Ako postoje sintaksičke greške, lokalizovane poruke kompajlera predstavljaju odziv sistema. Ako nema sintaksičkih grešaka, vrši se semantička provera korišćenjem invoke&test tehnike (Java reflexions). Ako postoje semantičke greške, studentsko rešenje upoređuje se sa sistemskim i tada se generišu poruke sistema u zavisnosti od rezultata upoređivanja. Ovu aktivnost vrši ekspertski modul koji koristi u pravilima ugrađeno domensko znanje. Zbog različitih tipova i težine problema, postoje specijalizovani podmoduli koji ispituju pojedinačne fragmente rešenja, a rezultate ispitivanja prikuplja glavni ekspertski modul, koji generiše odziv sistema prema studentu.

Primer korišćenja

Java PBL je dizajniran modularno – logika je odvojena potpuno od korisničkog interfejsa korišćenjem UML obrasca Fasada. Svi funkcionalni pozivi (API) realizuju se posredstvom ove klase, tako da je moguća saradnja između sistema. Kao samostalna Java PBL može da se koristi kao desktop ili Web aplikacija (JSP). U svim slučajevima koraci korišćenja su da student izabere sadržaj (vrstu problema). Sistem zatim generiše novi problem, koji student rešava. Rešenje zatim proverava sistem. Rezultat provere su primedbe sistema, prikaz tačnog rešenja i preporuke nastavnih materijala za učenje radi postizanja boljih rezultata.

Zaključak

Prikazan je jedan od načina kako da se omogući proširivanje funkcionalnosti LMS za PBL. Studenti na LMS sistemima usvajaju samo deklarativna znanja (definicije, objašnjenja, ilustracije, lekcije). Ova vrsta znanja (znati šta) nije dovoljna u rešavanju konkretnih domenskih problema (znati kako). Na drugoj strani student ne može da rešava probleme ako nema dovoljno deklarativnog znanja. To znači da su ovi sistemi komplementarni – da se nadopunjuju, a kvalitet učenja podiže se na viši nivo. Studentu su ponuđene obe vrste resursa, tako da može slobodno da ih koristi u skladu sa svojim stilom, tempom učenja i postavljenim zahtevima.

Java PBL ima otvorenu modularnu arhitekturu koja omogućava razne režime korišćenja, bilo kao samostalne aplikacije ili kao logički resurs za druge sisteme za e-učenje iz okruženja.

Ključne reči: *učenje rešavanjem problema, elektronsko učenje, inteligentni tutorski sistemi.*

Datum prijema članka: 26. 08. 2011.
Datum dostavljanja ispravki rukopisa: 17. 11. 2011.
Datum konačnog prihvatanja članka za objavljivanje: 18. 11. 2011.