Recommendation and Students' Authoring in Repositories of Learning Objects:

A Case-Based Reasoning Approach

doi:10.3991/ijet.v4s1.797

M. Gómez-Albarrán and G. Jiménez-Díaz
Universidad Complutense de Madrid, Madrid, Spain

***Abstract*—This paper presents a Case-Based Reasoning approach for the personalized recommendation and the students' authoring tasks in on-line repositories of Learning Objects (LOs). The recommender combines content-based filtering techniques together with collaborative filtering mechanisms. Students' authoring tasks include the incorporation of ratings of the existing LOs and new LOs, which are peer reviewed. This approach is going to be applied to a repository with more than 200 programming examples written in different programming languages.**

***Index Terms*—Case-based recommenders, Social filtering, Learning objects, Programming learning.**

I. INTRODUCTION

Skills in many different disciplines can only be developed by extensive practice. Instructors are conscious of that and they usually include many case studies and examples in their lectures. In the last years, this tendency has extended to the development of on-line repositories of Learning Objects (LOs), which represent these case studies and examples. The abundance of LOs inevitably poses a new challenge: providing support for locating those LOs adapted to the individual knowledge, goals and/or preferences of the students.

In this paper, we describe a Case-Based Reasoning (CBR) approach for recommending LOs that exist in educational repositories. This approach combines content-based filtering mechanisms and collaborative (or social) filtering processes [1]. The content-based filtering mechanisms take into account the student current knowledge and her concrete learning goals. The collaborative filtering mechanisms help to predict the utility that a concrete LO has for a student, based on the ratings (i.e., relevance, preferences, and opinions) that similar students (students with similar goals and knowledge level) have made about this LO.

Incorporating collaborative recommendation capabilities in an educational repository requires storing the rating score that a student makes about a LO together with her profile and learning goals. The student profile and goals are not static but evolve in time, so a collaborative filtering approach that considers the ratings made by similar students should store the profile and the goal the student had when she scored the LO. From a CBR point of view, the incorporation of this information represents a kind of learning in the approach. This new information will be used to refine the recommendation process. From the authoring point of view, the incorporation of this information represents the collaboration of a new agent: the student.

In our approach, students' authoring is not limited to the inclusion of rating scores about the existing LOs. Students can also suggest the incorporation of new LOs. So, the repository dynamically grows and improves with the student collaboration. Including all this information has three direct advantages. First, the content and organization of the repository is not limited to the instructor perspective but is complemented with the point of view of the students. Second, the development and updating of the repository puts a heavy load on the instructor, which can be reduced with the participation of the students. Finally, student motivation could increase because they collaborate in the learning process of their colleagues.

The general approach presented here will have a direct application in our daily teaching tasks at the Computer Science School in the Complutense University of Madrid (Spain). Programming skills greatly benefit from extensive practice and, as a consequence, many environments that support example-based programming teaching, as well as on-line repositories of programming examples, have been developed [2]. We have developed one of such repositories with more than 200 programming examples written in different programming languages. At present, this repository lacks of facilities to make personalized recommendations. Including these facilities could increase the current high use of the repository. Besides, we have suggested that the students are included in the authoring process. The suggestion has been well received by the students.

Section II provides a high-level description of the approach independently of the educational domain. Sections III and IV complements previous one by describing the recommendation and the authoring aspects. Section V briefly describes the particularization of the approach to an educational repository of programming examples. Section VI discusses our approach and relates it with previous works. Last section concludes the paper and presents some future directions for our research.

II. AN OUTLINE OF THE APPROACH

In this paper we propose a novel approach for managing LO repositories. This approach is characterized by the inclusion of two different interfaces:

- A reactive, single-shot recommender interface, which the students use to retrieve a set of relevant LOs after posing a query. The result is an ordered list of LOs. Priority is given to LOs that: are similar to the target query, adapt to the student knowledge level (student's profile), and were relevant to other students with a similar knowledge level.

- An authoring interface to create, review and rate new LOs. Students can add and catalogue a new LO according to the concepts that it covers. Moreover, they can also provide rating scores and comments for the new LOs created by their classmates.

The approach has been conceived as a CBR one [3]. CBR is a problem-solving paradigm that faces a new problem by retrieving past cases (experiences) that solve similar problems and reusing them in the new problem situation. CBR also is an approach to incremental learning that should include some maintenance policies [4]. Incorporating new cases to the case base is the essential way of learning in CBR. Learning retrieval knowledge is also a common way to improve the system performance.

CBR has been applied to diverse domains, from e-commerce applications to planning systems. In particular, CBR techniques have been successfully used for developing educational approaches and computer-based teaching systems [5, 6, 7] and case-based recommender systems [8].

From a CBR point of view the fundamental elements (the cases) of the knowledge base are the LOs contained in the repository. The cases should also enclose a set of associated rating scores and the student profile when the LO was scored. On the other hand, the recommendation task concerns to the approximate retrieval phase of the CBR, and the students' authoring interface relates to the CBR learning phase. We suggest the use of an ontology to index LOs within the repository. Ontologies provide a general indexing scheme that lets include similarity knowledge between concepts, which is a crucial knowledge in the similarity-based search and ranking contexts employed by the recommender. Besides, an ontology gathers a common parlance that can be used by the various knowledge sources (instructors, students) when including new LOs, and by the students when querying the tool. Other authors interested in collaborative authoring also make a successful use of ontologies [9].

Fig. 1 sketches our approach. Sections III and IV describe the components associated with the recommender and the authoring interfaces, respectively.

III. THE RECOMMENDER INTERFACE

The recommender interface provides access to the most relevant LOs for a student by posing explicit queries. Then, the recommendation runs in two stages: retrieval and ranking.

The retrieval stage follows knowledge-intensive CBR guidelines and looks for the LOs that satisfy, in an approximate way, the student learning goals. The student poses a query using the concepts existing in the domain ontology. This query represents her learning goals: the concepts she wants to learn. The 'retrieval component' tries first an exact match and finds the LOs indexed by the query concepts. If there are no LOs that satisfy this condition, LOs indexed by a subset of the (same or similar) concepts specified by the student are retrieved.

Once LOs are retrieved, the 'ranking component' sorts them according to the relevance assigned to each LO. The ranking stage mixes a compromise-driven selection strategy with user-based collaborative filtering to generate a relevance score for each LO [10]. Relevance score computes the relevance of the LO *L* for a student *S* as the sum up of two elements:

- An element that represents the compromise and combines:
  - o *The relevance due to the goals satisfied by L*. The higher the number of query concepts that *L* lets learn is, the higher the relevance value is. The more similar *L* concepts and query concepts are, the higher the relevance value is.
  - o *The relevance due to the adaptation degree of L to the current knowledge of S*. The student's current knowledge is represented by her profile in the 'student profile repository'. The goal is to penalize *L* if it includes concepts (different from those that let satisfy the query) that profile *S* lacks.
- An element that represents *the relevance due to the utility assigned to L by other students with goals and profiles similar to S*. The 'student preference repository' stores the rating scores from the students about the LOs they have used, together with their goals and profiles when they used them. We weight the rating scores from the other students according to the similarity between the profile when they ranked the LO and the current *S* profile. The more similar the profiles are, the higher the relevance for the rating score is.

In order to compute any of the partial relevancies, different local similarity metrics can be tried. In order to compute the global relevance we can choose among some of the widely accepted similarity metrics: (weighted) block-city, (weighted) Minkowsky, Euclidean, etc.

This case-based recommender alleviates most of the classic bottlenecks related to collaborative filtering recommender systems [11, 12, 13]. As we can see, the ontology-based indexing scheme is crucial for both the retrieval and ranking stages, and it reduces the *cold starting* and *first-rater* problems: even if a LO has only a few ratings or a student is new at the repository, the recommender provides a set of relevant LO according to the student query. Moreover, the user-based collaborative filtering strategy does not need to explore the complete student preference repository in order to look for similar students. Instead, the profiles are extracted from the ones that have already rated a candidate LO. This reduces the *scalability* problem that concerns to user-based collaborative filtering.

IV. STUDENTS' AUTHORING INTERFACE

The students' authoring interface lets learn two classical types of knowledge in CBR: cases and retrieval knowledge.

As stated before, the 'student preference repository' stores the rating scores explicitly assigned by the students to the LOs they have used, together with their goals and profile when they used them. According to Brusilovsky, Farzan and Ahn [14], we defend an explicit collection of student feedback because the information obtained is more accurate than the one obtained by implicit approaches

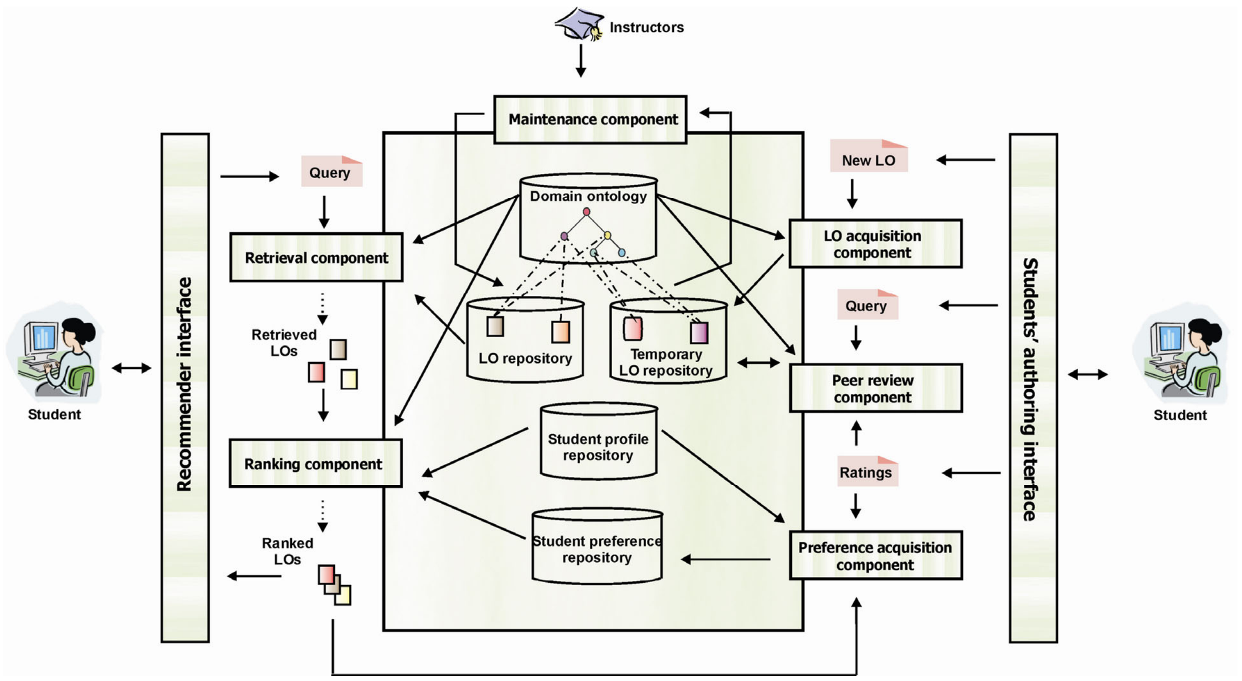

Figure 1. Sketching the approach.

which extract feedback from user actions. This way, the collected information is used by the recommender interface and provides a reliable form of collaboratively refining the relevance computed for each LO stored in the repository. So, the repository learns knowledge for the CBR retrieval phase as an indirect learning of index weights [15].

The 'LO acquisition component' and the 'peer review component' participate in the learning of new LOs. Students provide new LOs and a tentative set of the ontology concepts to index them. The 'LO acquisition component' stores them in a temporary LO repository until the instructor permanently moves them to the 'LO repository' using the 'maintenance component'. Once they are in the 'LO repository' they can be accessed.

The approach includes a manual CBR maintenance policy where instructors decide off-line which LOs they definitely incorporate into the 'LO repository'. A peer review technique allows students to examine and to judge the quality of the LOs provided by their colleagues. We propose that the 'peer review component' lets students browse the contents of the 'temporary LO repository' or pose a query to look for new LOs that relate to concepts they are interested in. The students give ratings, and optionally comments, about the new LOs, alleviating the classical *first-rater* problem in collaborative recommender systems. The information provided by the students will be taken into account by the instructor in the maintenance process.

Although we have suggested the use of a periodic, off-line and reactive maintenance policy aimed basically at LO retention, the approach also supports competence-preserving approaches to LO deletion that can profit from the rating scores assigned by the students. Low rating scores let identify redundant or uninteresting LOs, which the instructor could freely delete.

V. THE CASE OF A REPOSITORY OF PROGRAMMING EXAMPLES

In the last two years, a web repository of programming examples for CS1 and Physics students has been available through the Complutense University of Madrid Virtual Campus. The repository contains more than 200 programming examples developed using different programming languages. The examples are nowadays organized according to thematic packages and difficulty levels.

Although most of the students really appreciate the support provided by the repository, there is a 70% student that misses the facility to find the examples they are interested in. Certainly, this facility can not be provided by a course management tool like WebCT, which is used in our Virtual Campus. Our students are also very interested in participating in the authoring tasks: there is a 60% student that would like to participate by including their own examples and/or assessing existing examples.

In order to provide personalized recommendation and students' authoring facilities in this repository we are going to apply the approach described here. Examples will be indexed through an ontology of programming concepts that we have developed based on existing educational ontologies for procedural and object-oriented programming [16, 17]. The ontology has been designed using the ontology editor Protégé and formalized in OWL-DL [18]. Fig. 2 shows a partial view of our ontology.

The use of an indexing scheme based on programming concepts that exists in an ontology, instead of concrete programming instructions extracted from the programming code of the example solution, helps to separate the example representation from the programming style of the example author. On the other hand, our ontology is quite independent of the programming language. So it is possible to use it for indexing examples solved using different programming languages. This could be of great help when the student is

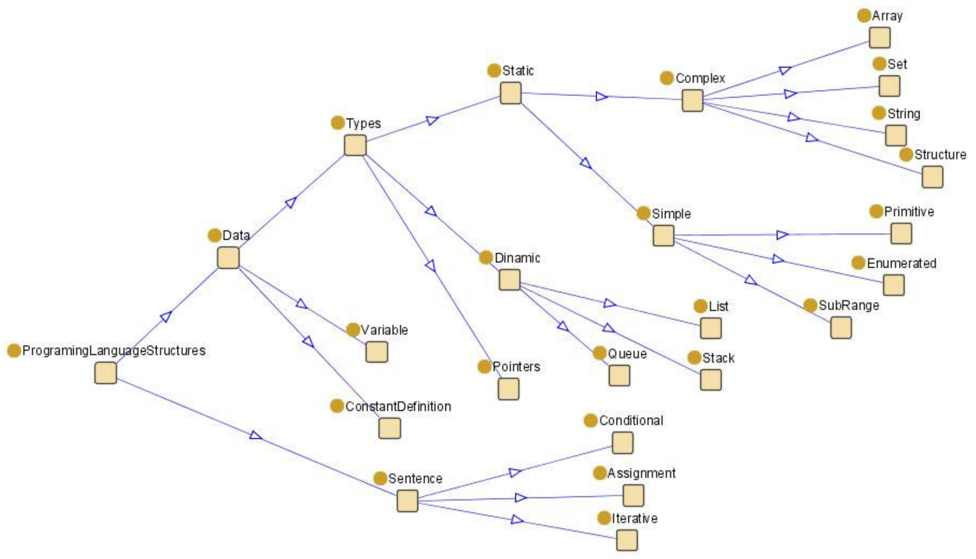

Figure 2. A partial view of the programming ontology.

interested in retrieving examples developed with different programming languages after posing a query.

In order to develop the prototype tool we will use jCOLIBRI-2 [19], an open source framework in Java that provides the main infrastructure to manage ontologies and implements the most common methods and comparison algorithms employed in case-based retrieval. Moreover, latest version of jCOLIBRI-2 includes templates to prototype case-based recommender systems.

The prototype tool will incorporate facilities for the automatic updating of the student model by using tests that exists for every aspect of the course. It will also distinguish among examples that are new to the student from those that she has previously accessed, as NavEx does [20]. This could help students to explore the examples retrieved.

We plan to include unsolved exercises into the LO repository. So, the student could retrieve completely solved examples and exercises to solve. When retrieving an exercise to solve the environment could make a personalized recovery of examples in the context of the exercise: examples would be requested by the student on demand or the environment could proactively suggest examples (i.e., if the solution of the exercise should include loop sentences and the student model shows a low knowledge of loops, the environment could suggest examples that include loops in their solution).

VI. RELATED WORK

Educational digital libraries rely mainly on content-based retrieval [14, 21]. LOs are considered as documents and search engines apply information retrieval methods, such as vector model space, to retrieve the LOs that satisfy the student query. Hybrid filtering approaches that combine content and collaborative aspects have been extensively used in e-commerce recommender systems [13]. However, they constitute a quite innovative approach in accessing educational repositories.

The approach described in the paper is inspired in several works that employ case-based retrieval methods to address classical problems from collaborative filtering. CASPER [22] is an online recruitment search engine that combines similarity-based reasoning and user profiling to provide personalized information retrieval. CASPER works also in two stages but it differs from our approach in the methods employed. On the one hand, retrieval stage employs feature-based similarity metrics. On the other hand, ranking stage employs only the rating profile from the user that made the query in order to rank the retrieved jobs.

PTVPlus [12] is a recommender for the Digital TV domain. In this case, it employs an item-based approach to suggest users' relevant TV programs. Although our approach promotes user-based collaborative filtering, PTVPlus has suggested us the use of CBR-like similarity metrics to compare user profiles and the relevance computing to rank the retrieved LOs.

Earlier example-based educational environments in the programming teaching domain had a very simple interface to select relevant examples and supported searching by using keywords that appear in the problem statement or in the code itself [23, 24]. These tools did not take the current student knowledge into account, so they could retrieve resources including concepts that the student does not know (or even that she is not ready to learn yet). Besides, it should be noticed that the problem statement could describe the example goal from very different points of view: it could include references to specific programming concepts –i.e., adding an element to a sorted linked-list– but it could describe an equivalent "real-world" problem –i.e., adding a new contact to an address book. Clearly, retrieval based on keywords when using this second (and common) type of example goal description is far from being appropriate, what clearly limits the use of the approach.

More recently, NavEx [20], an evolution of the web-based tool WebEx [25] for exploring annotated program examples developed using the C programming language, classifies examples according to the current state of the student knowledge and her history of past interactions. It applies adaptive navigation to: (a) distinguish new examples from examples that have already been partially or fully explored, and (b) categorize examples as being either "ready to" or "not yet ready to" explore according to the current knowledge of the student. Finally, NavEx ADVISE [26] extends the previous work on NavEx by combining adaptive annotation with spatial 2D similarity-based visualization. The whole repository of examples is displayed on a 2D example map where similar examples are placed closer to each other and dissimilar examples are placed farther from each other. Each example is represented by a vector of concepts from the C-programming domain. The concepts are automatically extracted from the code itself using a domain-specific parser and traditional information retrieval techniques. The concept vectors are used to calculate the similarities between the examples they represent. The use of automated textual indexing and retrieval methods can reduce the resource cataloging effort. However, our ontology-based approach let us include similarity metrics between concepts. This metrics are crucial for the retrieval stage and the compromise component in the rating stage.

NavEx ADVISE provides a limited support for locating resources adapted to the student current learning goal. Given an example currently explored by the student, she only receives visual cues about similar and dissimilar examples. However, the tool does not provide information about neither the concepts shared by two examples nor the concepts on which two examples differ. So, the student can not easily locate examples appropriate for a concrete learning goal.

Educational repositories that incorporate collaborative filtering employ approaches less accurate than the one proposed in our work. They take into account the LO ratings made by the students independently of their profile. This is the case of Knowledge Sea [14], a platform to access electronic documents about the C programming language that incorporates social navigation support. Knowledge Sea search engine uses a vector model rating to order the retrieved LOs according to their relevance but student ratings are not used to calculate the relevance. Instead, social relevance is expressed with visual cues.

VII. CONCLUSIONS AND FUTURE WORK

This paper presents a CBR approach for the personalized access and the students' authoring tasks in on-line repositories of LOs. The personalized access combines content-based filtering and collaborative user-based filtering mechanisms. Students could extend the repository with two different kinds of information: new LOs and their preferences about the existing LOs. So, the environment contents and behavior improve with the student collaboration. The use of an ontology-based indexing scheme is a crucial element in the approach. It provides a unique vocabulary for query retrieval and it eases the relevance metric computing according to the student's knowledge level.

In order to prove the feasibility of the approach described in this paper, we will apply it to an on-line repository with more than 200 programming examples appropriate for Computer Science and Physics non-major students. As we have shown, the use of techniques employed in case-based recommender systems in this domain introduces improvements with respect to related works. We plan to make a comprehensive evaluation of the resulting tool at the end of the next academic year.

Our immediate future work considers the evaluation and comparison of different similarity measures in order to compute the LO relevance and compromise according to the student's profile. Moreover, we should consider how to integrate other features included in the LO specification – for instance, difficulty level or programming language – in the similarity methods.

Finally, we also plan to integrate a proactive version of the recommender with learning management systems to generate personalized sets of LOs according to student assessments. In order to access to the following lecture, the student should pass a test. The student's grades in this assessment will be used to select the LOs that the environment provides to the student in the next lecture.

REFERENCES

[1] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating "word of mouth"", in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press, 1995, pp. 210-217.

[2] M. Gómez-Albarrán, "The teaching and learning of programming: A survey of supporting software tools", *Computer Journal*, vol. 48, n. 2, pp. 130-144, 2005. (doi:10.1093/comjnl/bxh080)

[3] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches", *AI Communications*, vol. 7, n. 1, pp. 39-59, March 1994.

[4] D.C. Wilson and D.B. Leake, "Maintaining case-based reasoners: dimensions and directions", *Computational Intelligence*, vol. 17, n. 2, pp. 196-213, May 2001. (doi:10.1111/0824-7935.00140)

[5] D.C. Edelson, "Learning from questions and cases: The Socratic case-based teaching architecture", *The Journal of the Learning Sciences*, vol. 5, n. 4, pp. 357-410, 1996. (doi:10.1207/s15327809jls0504_3)

[6] G. Jiménez-Díaz, M. Gómez-Albarrán, and P.A. González-Calero, "UnderFrame: Understanding Object-Oriented Frameworks Using a Case-Based Teaching Approach", in *Poster session at 18th European Conference on Object-Oriented Programming*, 2004.

[7] J. Kolodner, M.T. Cox, and P.A. González-Calero, "Case-based reasoning-inspired approaches to education", *Knowledge Engineering Review*, vol. 20, n. 3, pp. 299-303, September 2005. (doi:10.1017/S0269888906000634)

[8] D. Bridge, M.H. Göker, L. McGinty, and B. Smyth, "Case-based recommender systems", *Knowledge Engineering Review*, vol. 20, n. 3, pp. 315-320, September 2005. (doi:10.1017/S0269888906000567)

[9] J.M. Dodero, P. Díaz, I. Dodero, and A Sarasa, "Integrating ontologies into the collaborative authoring of learning objects", *Journal of Universal Computer Science*, vol. 11, n. 9, pp. 1568-1575, September 2005.

[10] R. Burke, "Hybrid web recommender systems", in *The Adaptive web*, Springer, 2007, pp. 377-408. (doi:10.1007/978-3-540-72079-9_12)

[11] A. Hinze, and S. Junmanee, "Advanced recommendation models for mobile tourist information", in *Proceedings of the OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006*, Springer, 2006, pp. 643-660.

[12] D. O'Sullivan, B. Smith, D.C. Wilson, K. McDonald, and A. Smeaton, "Improving the quality of the personalized electronic program guide", *User Modeling and User-Adapted Interaction*, vol. 14, n. 1, pp. 5-36, February 2004. (doi:10.1023/B:USER.0000010131.72217.12)

[13] K. Wei, J. Huang, and S. Fu, "A survey of e-commerce recommender systems", in *Proceedings of the International Conference on Service Systems and Service Management 2007*, IEEE Computer Society, 2007, pp. 1-5.

[14] P. Brusilovsky, R. Farzan, and J.W. Ahn, "Comprehensive personalized information access in an educational digital library", in *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, ACM, 2005, pp. 9-18.

[15] P. Gomes and C. Bento, "Learning user preferences in case-based software reuse", in *Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning*, Springer-Verlag, 2000, pp. 112-123.

[16] N. Henze, N., P.Dolog, W. Nejdl, "Reasoning and ontologies for personalized e-learning in the semantic web", *Journal of Educational Technology & Society*, vol. 7, n. 4, pp. 82-97, October 2004.

[17] S. Sosnovsky and T. Gavrilova, "Development of educational ontology for C-programming", *International Journal Information, Theories & Applications*, vol. 13, n. 4, 2006, pp. 303-307.

[18] OWL Website: http://www.w3.org/TR/owl-features (last access: January 6th, 2009).

[19] JColibri-2 Website: http://gaia.fdi.ucm.es/grupo/projects/jcolibri/ (last access: January 6th, 2009).

[20] M. Yudelson and P. Brusilovsky, "NavEx: Providing navigation support for adaptive browsing of annotated code examples", in *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, IOS Press, 2005, pp. 710-717.

[21] C. Lagoze, W. Arms, S. Gan, D. Hillmann, C. Ingram, D. Krafft, R. Marisa, J. Phipps, J. Saylor, C. Terrizzi, W. Hoehn, D. Millman, J. Allan, S. Guzman-Lara, T. Kalt, "Core services in the architecture of the national science digital library (NSDL)", in *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, ACM, 2002, pp. 201-209.

[22] K. Bradley and B. Smith, "Personalized information ordering: a case study in online recruitment", *Knowledge-Based Systems*, vol. 16, n. 5, pp. 269-275, July 2003. (doi:10.1016/S0950-7051(03)00028-5)

[23] J.M. Faries and B.J. Reiser, "Access and use of previous solutions in a problem solving situation", in *Proceedings of the Annual Conf. of the Cognitive Science Society*, Laurence Erlbaum Associates, 1988, pp. 433-439.

[24] L.R. Neal, "A system for example-based programming", *SIGCHI Bulletin*, vol. 20, n. SI, pp. 63-68, 1989.

[25] P. Brusilovsky, "WebEx: Learning from examples in a programming course", in *Proceedings of the WebNet'2001, World Conference of the WWW and Internet*, AACE, 2001, pp. 124-129.

[26] P. Brusilovsky, J.W. Ahn, T. Dimitriu, and M. Yudelson, "Adaptive knowledge-based visualization for accessing educational examples", in *Proceedings of the 10th International Conference on Information Visualization*, IEEE Computer Society, 2006, pp. 142-150.

AUTHORS

M. Gómez-Albarrán is with the Department of Ingeniería del Software e Inteligencia Artificial, Complutense University of Madrid, Madrid, 28040, Spain (e-mail: albarran@ sip.ucm.es).

G. Jiménez-Díaz is with the Department of Ingeniería del Software e Inteligencia Artificial, Complutense University of Madrid, Madrid, 28040, Spain (e-mail: gjimenez@fdi.ucm.es).

This work has been supported by the Spanish Committee of Education and Science project TIN2006-15202-C03-03, and also by the UCM project PIMCD2008-136.

This article was modified from a presentation at X International Symposium on Computers in Education (SIIE2008) 1st-3rd October 2008, Salamanca, Spain. Manuscript received 12 January 2009. Published as submitted by the authors.