Personalized Access and Students' Coauthoring in Repositories of Learning Objects: The Case of a Repository of Programming Examples[*]

Mercedes Gómez-Albarrán
Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
albarran@sip.ucm.es

Susana Bautista-Blasco Jorge Carrillo de Albornoz
Facultad de Informática
Universidad Complutense de Madrid
{sbblasco, jorge.carrillodealbornoz@gmail.com}

Abstract

This paper presents a Case-Based Reasoning approach for the personalized access and the students' coauthoring tasks in on-line repositories of Learning Objects (LOs). The personalized access combines content-based filtering techniques together with collaborative filtering mechanisms. Students' coauthoring tasks include the incorporation of: assessments of the existing LOs, and new LOs which are peer reviewed. The approach has been applied to a repository with more than 200 programming examples written in different programming languages.

1. Introduction

Programming skills can only be developed by extensive practice. Instructors are conscious of that and include many program examples in their lectures. In the same way, environments that support example-based programming teaching, and on-line repositories of programming examples, have been developed [1].

The abundance of programming LOs available poses a new challenge: providing support for locating those adapted to the individual knowledge, goals and/or preferences of the students.

Earlier educational example-based environments 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 [2] [3]. 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 including concepts 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 [4], an evolution of the web-based tool WebEx [5] 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 [6] 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, which are automatically extracted from the code itself using a domain-specific

[*] This work has been partially supported by the Spanish Committee of Education and Science project TIN2006-15202-C03-03 and the Comunidad de Madrid Education Council and Complutense University of Madrid (consolidated research group 910494).

parser and traditional information retrieval techniques. The concept vectors are used to calculate the similarities between the examples they represent.

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 provides 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.

In this paper, we describe a general approach for providing personalized access to educational repositories that adapts to both, the student current knowledge and her concrete learning goal. This content-based filtering mechanism is supplemented with a collaborative filtering process [7] that helps to predict the utility that a concrete LO has for a new student, based on the assessments (i.e., relevance, preferences, opinions) that similar students (students with similar goals and knowledge level) have made about this LO. The approach has been applied to a repository where the LOs are programming examples.

Hybrid filtering approaches that combine content and collaborative aspects have been used in recommender systems [8]. But they constitute a quite innovative approach in accessing educational repositories. Educational digital libraries rely mainly on content-based retrieval [9]. Educational repositories that incorporate collaborative filtering employ approaches less accurate than the one proposed in our work. They take into account the LO assessments made by the students independently of their profile. This is the case of Knowledge Sea [10], a platform to access electronic documents about the C programming language that incorporates social navigation support.

Our proposal of incorporating collaborative recommendation capabilities in an educational repository requires storing the assessment that a student makes about a certain LO together with his profile and learning goal (notice that student profile and goals evolve in time, so a collaborative filtering approach that considers the assessment made by *similar* students should store the profile and the goal the student had when she assessed the LO). This information can be considered as an extension of the initial knowledge stored and represented by the instructors that create the repository, which will be used to refine the access mechanism and the searching results. In this sense, the incorporation of this information can be considered as a part of a coauthoring process of the students.

Finally, another important aspect of our approach is that the student coauthoring process also considers the incorporation of new LOs. So, the repository dynamically grows and improves with the student collaboration.

Including all this information has two direct advantages. On the one hand, 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. On the other hand, student motivation could increase because they collaborate in the learning process of their colleagues.

Section 2 presents the approach independently of the educational domain. Section 3 briefly describes the particularization of the approach to an educational repository of programming examples. Last section concludes the paper and presents future work.

2. A General View of the Approach

Figure 1 shows the architecture that supports our approach. Subsections 2.1 and 2.2 describe the components associated with the personalized access and the coauthoring process that involves the students.

The approach has been conceived as a Case-Based Reasoning (CBR) one [11]. 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 [12]. 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 [13][14][15].

From a CBR point of view the LOs of the repository are the fundamental elements (the cases) of the knowledge base, the personalized access corresponds to the approximate retrieval phase of the CBR, and the students' coauthoring tasks relate to the CBR learning phase. We suggest the use of an ontology-based indexing scheme of the LOs instead of a textual indexing (such as the one used in the tools described in Section 1 or in some educational digital libraries). The use of automated textual indexing and retrieval methods can reduce the resource cataloging effort. However, ontology-based cataloging provides a general indexing scheme that lets include similarity knowledge between concepts, which is a crucial

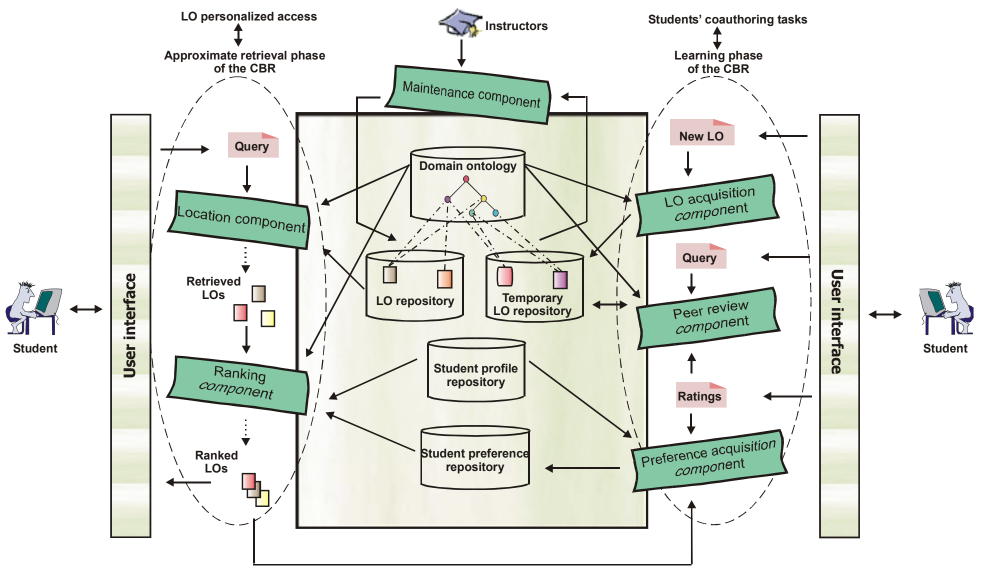

Figure 1. The architecture that supports the approach.

knowledge in similarity-based search and ranking contexts. 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.

2.1. Personalized access

According to the CBR paradigm, access is organized in two steps: location and ranking.

The location step finds the LOs that satisfy, in an approximate way, the student learning goal. The student poses a query to the tool by using the concepts existing in the domain ontology. This query represents her learning goals: the concepts she likes to learn. The LOs indexed by the query concepts are retrieved. 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, they are ranked according to the relevance assigned to each LO by a similarity metric. Similarity metric computes the relevance of the LO *X* as the sum up of three elements, which are intuitively described next (technical details of the metric are out of the scope of the paper):

(a) The relevance due to the goals satisfied by *X*. The higher the number of query concepts that *X* lets learn is, the higher the relevance value is. The more similar *X* concepts and query concepts are, the higher the relevance value is.

(b) The relevance due to the adaptation degree of *X* to the student current knowledge (represented by her profile in the 'student profile repository'). The objective is to penalize *X* if it includes concepts (different from those that let satisfy the query) that are unknown to the student.

(c) The relevance due to the utility assigned to *X* by other students with goals and profiles similar to the current one. The 'student preference repository' stores the assessments made by the students about the LOs they have used, together with their goals and profiles when they used them. It is not necessary to explore the complete student preference repository in order to look for *similar* students, but they are looked for among the students that have valued *X*. This reduces the classic bottleneck related to collaborative filtering in recommender systems.

As we can see, the ontology-based indexing scheme is crucial for both, the location and the ranking steps.

2.2. Students' coauthoring tasks

From the CBR point of view, the students' coauthoring facilities considered in our approach let learn two types of knowledge: cases and retrieval knowledge.

As stated before, the 'student preference repository' stores the assessments explicitly assigned by the students to the LOs they have used, together with their goals and profile when they used them. We agree with Brusilovsky *et al.* [10] and defend an explicit collection of student feedback because the information

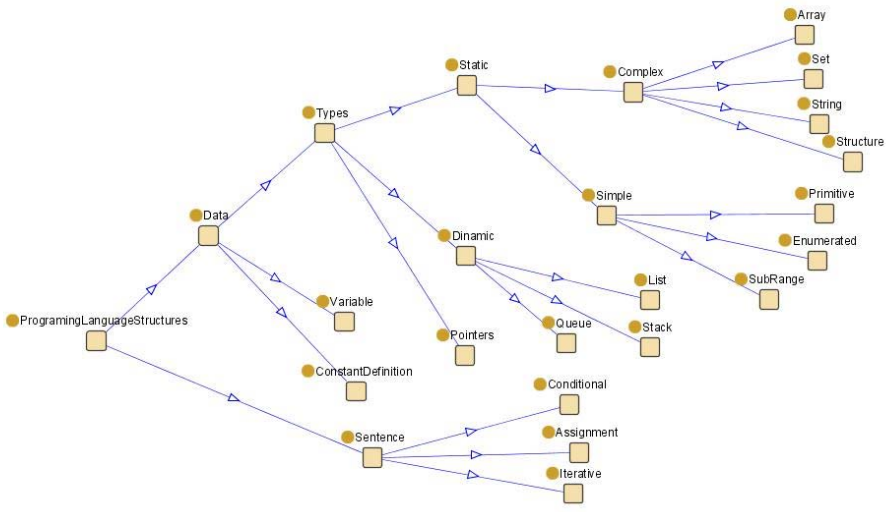

Figure 2. A partial view of the programming ontology (obtained with Protégé facilities).

obtained is more accurate than the one obtained by implicit approaches which extract feedback from user actions. This way, the collected information is used by the 'ranking component' in the personalized access process and provides a reliable form of collaboratively refining the relevance computed for each LO located. So, the tool learns knowledge for the CBR retrieval phase as an indirect learning of index weights [16].

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.

We have chosen a manual CBR maintenance policy where instructors decide off-line which LOs they definitely incorporate into the 'LO repository'. Once more, the students' point of view complements the instructor perspective. A peer review technique allows students to examine and to judge the quality of the LOs provided by their colleagues. 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 can give ratings, and optionally comments, about the new LOs that will be taken into account by the instructor in the maintenance process.

3. Applying the Approach to a Repository of Programming Examples

The approach described here is nowadays followed to provide personalized access and students' coauthoring facilities in a repository of more than 200 programming examples developed using different programming languages (C++, Java, Pascal). The examples were originally developed to support introductory programming courses in Computer Science and Physics at the Complutense University de Madrid. Students will use the tool in the second semester of this academic year.

Examples are indexed through an ontology of programming concepts that we have developed based on existing educational ontologies for procedural and object-oriented programming [17][18][19]. The ontology has been designed using the ontology editor Protégé and formalized in OWL-DL [20]. Figure 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 based on 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 interested in retrieving examples developed with different programming languages after posing a query.

The tool will incorporate student model update facilities. The student model will be automatically updated after the evaluation of the tests that exists for every aspect of the course.

We will try the inclusion of 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 tool could make a personalized recovery of examples in the context of the exercise: examples would be requested by the student on demand or the tool could suggest examples of its own accord (i.e., if the solution of the exercise should include loop sentences and the student model shows a low knowledge of loops, the tool could suggest examples that include loops in their solution).

4. Conclusions and future work

This paper presents an approach *à la* CBR for the personalized access and the students' coauthoring tasks in on-line repositories of LOs. The personalized access combines content-based filtering and collaborative filtering mechanisms. Students could extend the tool knowledge with two different kinds of information: new LOs and their preferences about the existing LOs. So, the tool contents and behavior improve with the student collaboration. The use of an ontology-based indexing scheme is a crucial element in the approach.

The approach is applied to an on-line repository with more than 200 programming examples appropriate for Computer Science and Physics non-major students. As Section 1 shows, the tool introduces improvements with respect to the related works.

Our immediate future work considers three aspects. We plan to study the inclusion of automatic CBR maintenance policies in the general approach, so instructors are free from maintenance tasks. We will start the extension of the tool with exercises. And, finally, we also plan to make a comprehensive evaluation of the tool at the end of the second semester of this academic year.

5. References

[1] M. Gómez-Albarrán, "The Teaching and Learning of Programming: A Survey of Supporting Software Tools", *The Computer Journal*, 48(2), 2005, pp. 130-144.

[2] L.R. Neal, "A System for Example-Based Programming", *Proc. Human Factors in Computing Systems Conference*, ACM Press, New York, 1989, pp. 63-68.

[3] J.M. Faries, and B.J. Reiser, "Access and Use of Previous Solutions in a Problem Solving Situation", *Proc. Annual Conf. of the Cognitive Science Society*, Laurence Erlbaum Associates, New Jersey, 1988, pp. 433-439.

[4] M. Yudelson, and P. Brusilovsky, "NavEx: Providing Navigation Support for Adaptive Browsing of Annotated Code Examples", *Proc. Int. Conf. on Artificial Intelligence in Education*, IOS Press, Amsterdam, 2005, pp. 710-717.

[5] P. Brusilovsky, "WebEx: Learning from Examples in a Programming Course", *Proc. World Conference of the WWW and Internet*, AACE Press, New York, 2001.

[6] P. Brusilovsky, J. Ahn, T. Dumitriu, and M. Yudelson, "Adaptive Knowledge-Based Visualization for Accessing Educational Examples", *Proc. Information Visualization Conference*, IEEE, New York, 2006, pp. 142-147.

[7] U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating "Word of Mouth"", *Proc. Conf. on Human Factors in Computing Systems*, vol. 1, ACM, New York, 1995, pp. 210-217.

[8] K. Wei, J. Huang, and S. Fu, "A Survey of E-Commerce Recommender Systems", *Proc. Int. Conf. on Service Systems and Service Management*, 2007, pp. 1-5.

[9] C. Lagoze, W. Arms, S. Gan, *et al.*, "Core services in the architecture of the national science digital library (NDSL)", *Proc. ACM/IEEE-CS Conference on Digital libraries*, ACM, New York, 2002, pp. 201-209.

[10] P. Brusilovsky, R. Farzan, and J. Ahn, "Comprehensive Personalized Information Access in an Educational Digital Library", *Proc. ACM/IEEE-CS Conf. on Digital Libraries*, ACM, New York, 2005, pp. 9-18.

[11] A. Aamodt, and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", *AI Communications*, 7(1), 1994, pp. 39-59.

[12] D.C. Wilson, and D.B. Leake, "Maintaining Case-Based Reasoners: Dimensions and Directions", *Computational Intelligence*, 17(2), 2001, pp. 196–213.

[13] D.C. Edelson, "Learning from Questions and Cases: The Socratic Case-Based Teaching Architecture", *The Journal of the Learning Sciences*, 5, 1996, pp. 357-410.

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

[15] J.L. Kolodner, M.T. Cox, and P.A. González-Calero, "Case-Based Reasoning-Inspired Approaches to Education", *The Knowledge Engineering Review*, 20(3), 2005, pp. 299-304.

[16] P. Gomes, and C. Bento, "Learning User Preferences in Case-Based Software Reuse", *Proc. European Workshop on Case-Based Reasoning*, Springer, Berlin, 2000, pp. 112-123.

[17] S. Sosnovsky, and T. Gavrilova, "Development of Educational Ontology for C-Programming", *Information Theories & Applications*, 13(4), 2006, pp. 303-307.

[18] Java Ontology of AES Personal Reader for eLearning: http://personal-reader.de/rdf/java_ontology.rdf.

[19] C Programming Ontology: http://www.sis.pitt.edu/~paws/ont/c_programming.rdfs.

[20] OWL web site: http://www.w3.org/TR/owl-features.