Translation of Overlay Models of Student Knowledge for Relative Domains Based on Domain Ontology Mapping

Sergey Sosnovsky[1], Peter Dolog[2], Nicola Henze[3], Peter Brusilovsky[1], Wolfgang Nejdl[3]
[1]*School of Information Sciences, University of Pittsburgh, USA*
[2]*Department of Computer Science, Information Systems Unit, Aalborg University, Denmark*
[3]*L3S Research Center, University of Hannover, Germany*
sas15@pitt.edu, dolog@cs.aau.dk, henze@l3s.de, peterb@pitt.edu, nejdl@l3s.de

Abstract. The effectiveness of an adaptive educational system in many respects depends on the precision of modeling assumptions it makes about a student. One of the well-known challenges in student modeling is to adequately assess the initial level of student's knowledge when s/he starts working with a system. Sometimes potentially handful data are available as a part of user model from a system used by the student before. The usage of external user modeling information is troublesome because of differences in system architecture, knowledge representation, modeling constraints, etc. In this paper, we argue that the implementation of underlying knowledge models in a sharable format, as domain ontologies – along with application of automatic ontology mapping techniques for model alignment – can help to overcome the "new-user" problem and will greatly widen opportunities for student model translation. Moreover, it then becomes possible for systems from relevant domains to rely on knowledge transfer and reuse those portions of the student models that are related to overlapping concepts.

Introduction

Adaptive educational systems (AES) have a long history of research studies that can be traced back to early 70-s [1]. Over the years, various types of AESs [2] have demonstrated their effectiveness. The central component of every AES is a student model (SM), where the system stores its beliefs about the different characteristics of a student. The proximity of modeling assumptions to the actual state of a student determines how adequate the system's adaptive interventions can be. One of the problems shared by all adaptive systems is the "*new-user"* problem. The models of those students who have just started their work with the system are inherently poor, since the system has little or no evidence from which to draw its assumptions.

There are three traditional approaches to dealing with the "new-user" problem in modeling student knowledge. The majority of the systems use the so-called *tabula-rasa* (clean slate) approach, i.e., they assume that the user has no knowledge of the subject. Some systems begin, instead, with a reasonably sized questionnaire, aimed at assessing the user's starting level of knowledge and thus seed the user model accordingly, e.g., ELM-ART [3]. Systems supporting editable student modeling allow students to enter their knowledge levels based on intuition and previous domain experience [4]. Neither of these approaches is perfect. The first one is not appropriate in situations where new users are likely to have some previous reasonable knowledge of the subject. The second consumes time and intervenes with the educational process. Finally, editable SMs rely on subjective opinion of a student, which might be influenced by irresponsible or biased judgment and inability to correctly assess own knowledge.

Our paper suggests an alternative approach: to initialize the SM of one AES based on an existing model of this student built by another AES. This approach avoids time-consuming questionnaires and potentially biased self-evaluation while attempting to achieve a good quality of student modeling from the very beginning. Such scenarios have become quite realistic nowadays. As the WWW has become the ultimate platform for information and service delivery, numerous Web-based AESs have appeared [2]; hence it is possible to expect that the same student would interact with several AESs during his/her course of study. Translation (or mediation) of user models from one system to another is an emerging research field of user modeling (see for example, [5], [6]). Unfortunately, the design of most AESs does not support model sharing. Several teams are working on the development of distributed architectures for online learning, where AESs operate as components, using the same central UM server, e.g., ActiveMath [7], Personis [8]. However, these

systems cannot be easily plugged into a different architecture because of the discrepancies in communication protocols and model representation.

The problem of SM mediation becomes even harder if we consider mediation beyond a particular domain, e.g., between C and Java programming. Such domains share a large number of concepts; therefore, for a student studying C programming, a knowledge transfer [9] to Java should occur. Let's imagine that our student has been studying C programming with the help of a standalone AES, which collected an overlay model [10] of his/her knowledge. Now the student is planning to use another system for learning Java. In this situation the *tabula-rasa* SM of the AES teaching Java would not reflect the actual state of knowledge s/he has. As a result, the adaptation performed by the Java-system will not be adequate; it can even be potentially harmful. We could avoid this by using information from the SM of the C-system to initialize relevant parts of the SM representing Java knowledge (see fig. 1). However, first, it would be necessary to translate this information so that the second system would be capable of understanding it.

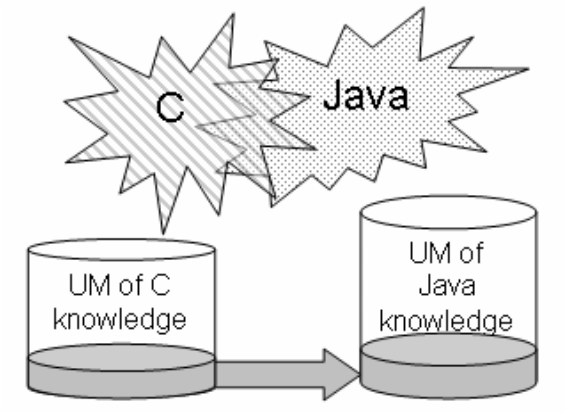

Figure 1. The relationship between C and Java knowledge.

We propose a solution to the described problem, based on the technologies that have recently become available in the framework of the Semantic Web initiative [11]. The implementation of the AES's domain model in a sharable format, as a web-ontology, enables access to this model by other AES. The ontology mapping techniques [12] help to automatically identify similar concepts in the ontologies of related domains and align the domain models of the two AES. Once found, the mapping can be used as a template for the translation of overlay SMs from one system to another. The idea to apply ontologies for knowledge representation in AESs is not new [13], [14]. Several attempts have been made towards implementation of ontology-based interoperable SMs. Denaux et al. [15] developed Onto-AIMS, a system integrating two AESs using a shared ontology. Authors of [16] describe Medea, a learning portal supporting cross-application adaptation on the basis of manual mapping of ontologies underlying a particular AES to the central domain ontology. Mitrovic and Devedzic [17] propose M-OBLIGE – an architecture for ITS interoperability based on the central ontology used as a reference point by the local ITS ontologies. Unlike these projects our approach does not imply a specific ontology shared my multiple AESs or a core system, used as a central interpreter by others. We investigate the possibility of using general ontology mapping techniques to *automatically* translate between SMs based on *arbitrary* ontologies from *relative* domains sharing a set of concepts. To validate our hypotheses, we have conducted an experiment in an introductory Java class. The statistical analysis demonstrates that automatic ontology mapping provides a solid basis for translation of overlay SMs between relative domains.

The rest of the paper is structured as follows. Section 1 describes the ontologies and educational content models, as well as gives the details of the mapping algorithm. Section 2 analyzes the results of the conducted experiment. Section 3 presents further discussion questions and sketches the directions of future work. Finally, section 4 concludes the paper.

1. Ontology Mapping for Translation of Student Models

We assume here that the systems whose domain models are being aligned follow basic principles of the formal representation of domain knowledge by ontologies. However, even when the AESs operate on similar domains, their domain models usually differ in structure, concept labels, granularity and modeling focus, as well as in the indexing and instantiation of educational content. Therefore, as a basis for ontology mapping, we chose an algorithm which assesses concept similarity based on their instances, using machine learning (ML) techniques. Such algorithms seem to be more robust towards managing the modeling discrepancies listed

above. Instead of the intended meaning of the concepts being mapped (*what is the concept?*) they concentrate on the operational meaning (*what is the concept about?*). The input data for any ML-based ontology mapping algorithm are the two ontologies, along with the sets of instances corresponding to the concepts of these ontologies. In the framework of AES, an instance of a concept is an occurrence of this concept in a learning resource. The next subsection describes the ontologies we mapped, along with their accompanying instances.

1.1. Domain Ontologies and Content Modeling

In our experiment we used two existing ontologies – one for Java and one for C. The Java Ontology was developed for the AES *Personal Reader for eLearning* [18]. The ontology is represented in RDFS and can be accessed at *http://personal-reader.de/rdf/java_ontology.rdf*. It defines 490 *rdfs:Class*'s connected to each other by the generic relation *rdfs:subClassOf*. The Java Ontology has been used to index the online SUN Java Tutorial (*http://72.5.124.55/docs/books/tutorial/java/index.html*), which primarily has been used as a resource for developing this ontology. The description of the tutorial structure and the index of its pages in terms of the concepts of the Java Ontology is represented as another RDF document [18].

The C Programming Ontology was also developed in RDFS. It was designed at the University of Pittsburgh and can be accessed at *http://www.sis.pitt.edu/~paws/ont/c_programming.rdfs*. The ontology defines 575 *rdfs:Class*'s and 5 *rdf:Property*'s. Table 1 summarizes some basic characteristics of the two ontologies. The Java Ontology covers a much bigger domain (including knowledge of object-oriented programming and some parts of the Java standard library); therefore the percentage of manually-mapped concepts is much smaller for Java than it is for C. At the same time, the C Programming Ontology provides a finer-grained representation of the domain's conceptual structure. During the manual mapping, we found many situations where one Java concept mapped to several C concepts; for example, the manual mapping contains more C concepts than Java concepts.

Table 1. Basic metrics of the two ontologies.

	Java Ontology	**C Programming Ontology**
Number of rdfs:Class's (concepts) defined	490	575
Number of concepts mapped	108 (22%)	257 (45%)
Number of rdf:Property's defined	0 (uses rdfs:subClassOf)	5 (isA, partOf, directPartOf, implement, utilize)
Number of RDF-triples	1013	1538

The focus of modeling for the C Programming Ontology was also different. While the Java Ontology was initially created to describe the content of the SUN tutorial, the C Programming Ontology was deliberately developed as a general-purpose ontology, in an attempt to model the domain of C programming in a maximally precise and unbiased way. These differences in modeling perspectives added additional difficulties for mapping. Figure 2 visualizes an extract from the C Programming Ontology. The content for indexing by the C Programming Ontology was taken from two online C tutorials: University of Leicester C Tutorial (*http://www.le.ac.uk/cc/tutorials/c/*) and Rob Miles's Tutorial (*http://www.eumus.edu.uy/eme/c/c-introduction_miles/contents.html*).

1.2. Ontology Mapping Algorithm

To perform ontology mapping, we implemented a modified version of the GLUE approach [19]. This algorithm computes the matrix of similarities between all concepts in two ontologies. The main idea of GLUE is based on the cross-classification of concept instances, counting of instances associated with concepts from different ontologies and following the calculation of joint probabilities for each pair of concepts. GLUE was chosen for its clear and robust algorithm, which could be easily adjusted for our purposes. A tutorial page indexed with a concept is seen as its instance. Pages were classified based on their term vectors. Term vector calculation included stemming (with the help of Porter stemmer), stop word filtering and TFIDF computation.

The quality of the GLUE algorithm is determined by the precision of the classifiers, which depends highly on the quality and size of the training data sets. To reduce this dependency, we added an extra mapping step based on the concept names. In this step, concept names are now parsed into separate words and compared based on lexical similarity. To calculate the lexical similarity of two words, we used Levenshtein's edit distance [20] and the metric proposed in [21]. The output of name-based mapping is used as the input for instance-based mapping.

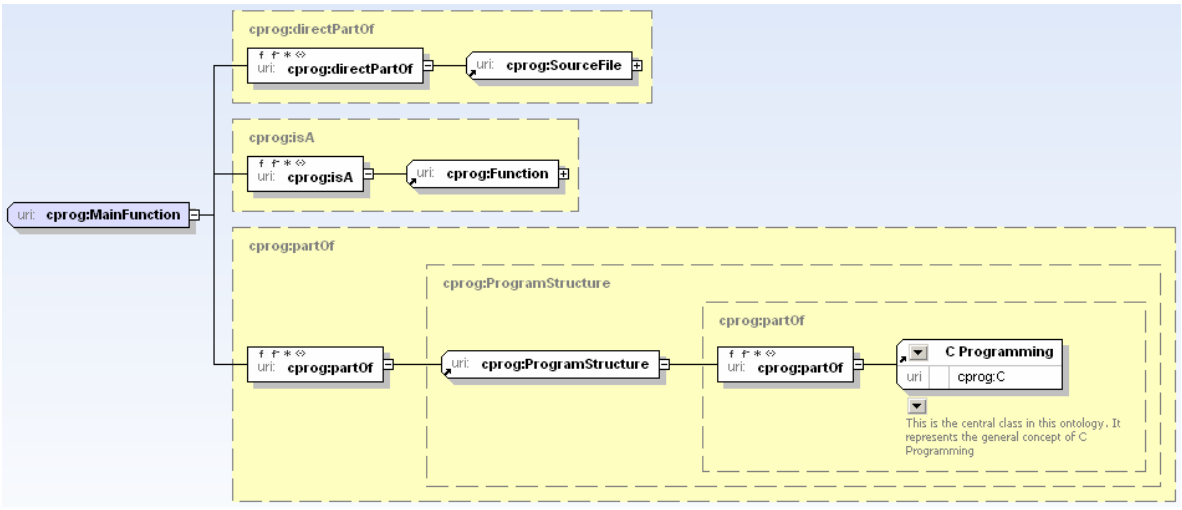

Figure 2. An extract from the C Programming Ontology.

Another characteristic of the input data, which has had a negative impact on the mapping precision, was the large percentage of instances shared by several concepts (i.e., tutorial pages are indexed with a number of concepts). This situation is typical for adaptive content authoring, but not for ontological engineering. As a result, the mapping algorithm could not distinguish between close concepts that often occur together (e.g. *StatementIf* and *RelationalOperator*). To deal with this problem, when training a classifier for a concept, the mapping algorithm weights term vectors according to the degree of influence that the concept has on this tutorial page. Hence a page indexed with a set of *n* concepts, including the concept *A,* contributes to the training of the classifier for this concept *n* times less then a page whose index contains only the concept *A*.

The original GLUE algorithm was developed to find the best possible "1-to-1" mappings for all concepts, from the two ontologies. In our algorithm, we relaxed this requirement, since our main goal was not the ontology mapping itself, but student knowledge translation based on the found mapping of domain ontologies. Sometimes the mapping algorithm is unable to distinguish between closely-related concepts because they have similar names and definitions, as well as share many instances. However, this is less important in the context of knowledge modeling, because students tend to have similar knowledge for related concepts (e.g., *WhileLoop* and *DoWhileLoop*). Hence, instead of finding the best "1-to-1" mapping, our algorithm finds several best candidates for every concept, where mapping is possible. These candidates are weighted according to the similarity values found during the mapping process. When SMs are translated these weights determine the percentage of contribution by candidate concepts from the first ontology to the resulting knowledge level of the target concept from the second ontology. This feature helps deal with the "1-to-many" mappings, especially when the candidate concepts are from different parts of the ontology. For example, good candidates for the concept *return* from the Java Ontology would be concepts *FunctionReturnValue* and *StatementReturn*, which belong to different branches of the C Programming Ontology.

2. The Experiment

2.1. Data Collection

The experiment was conducted in an introductory course on Object-oriented programming, which was being taught to undergraduate students in the School of Information Sciences, at the University of Pittsburgh. At the beginning of the course, we collected a questionnaire assessing their initial experience in C and Java programming. Out of 36 students, seven demonstrated a prior knowledge of Java and therefore were not eligible for the main experiment (we will call this group the "*Java"-group*). Eight students knew neither C nor Java; they formed the control group for the evaluation of C-model quality (we call it the *"¬C ¬Java"-group*). Finally, 21 students had C programming experience but had never used Java before. They became our main experimental group (*"C ¬Java"-group*). During the analysis of outliers, six cases were filtered out: three by their Java test scores and three by their C test scores. The final number of students in the groups was respectively 6, 7 and 17.

The 10-question pre-quiz was used for the initialization of the SM for C programming. As a result, we estimated the prior knowledge of 37 concepts from our C ontology. To evaluate Java knowledge, we used weekly quizzes. Each of these quizzes included 2-3 extra-credit questions checking the student's knowledge of several Java concepts that have some analogy in C language. At the time of the quiz these concepts were not yet mastered in class, so the recorded knowledge of these concepts was a result of transfer from C, not the newly acquired

knowledge. This gave us overlay models of the knowledge of 21 concepts of the Java Ontology. The problems (both in C pre-quiz and in Java extra-credit questions) assessed the students' knowledge on the level of application – they had to analyze code fragments and answer related questions.

To create the "*ground truth*" for our automatic ontology mapping experiment, we performed the manual mapping of these sets and identified 17 mapping cases from four large categories: *Functions/Methods*, *Operators*, *Simple Data Types* and *Control Structures*. All 17 cases were "1-to-1" mappings. They were part of the best mappings one could achieve between these two ontologies, for which we had overlay models of student knowledge in both C and Java. The automatic ontology mapping reduced this subset to nine concepts. It became our target set of concepts. For these nine concepts we were able to compare the results of the SM translation, based on the automatic ontology mapping with the translation being based on the manual ("*perfect*") mapping.

2.2. SM Validation

The results of the questionnaire on programming experience (the size of the *"C ⌐Java"-group*) confirmed the scenario described in the introduction, i.e., the presence of a sufficient number of students moving from C to Java. However, the problem for our experiment was to find enough students of that kind *with the populated SM of C knowledge*. This goal was hard to accomplish in the scale necessary for statistical data analysis. Therefore, the C-knowledge SMs have been initialized using the pre-quiz. To verify the results of the pre-quiz and consequently to validate collected SMs, we used the data from the programming experience questionnaire.

The population of students who did not know Java consisted of both the *"⌐C ⌐Java"-group* (7 subjects) and the *"C ⌐Java"-group* (17 subjects). For all students, we calculated the cumulative level of C knowledge based on the results they showed on the pre-quiz for the target set of concepts. The necessary assumptions of parametric statistical tests were not met. The distributions of data across both groups were severely skewed from the normal curve and the data sample variances were not homogeneous as well. Therefore, to compare the mean scores of these groups we employed the Wilcoxon-Mann-Whitney test as a t-test substitute. The mean score for students with C experience (*0.94 ± 0.01*) was significantly greater then the mean knowledge level for students who were novices in C programming (*0.64±0.11*): *Mann-Whitney U statistics = 19.00; p = 0.010*. Table 2 summarizes the results of statistical analysis.

Similarly, we estimated the validity of the Java SMs. The average score for the mapped subset of Java concepts was calculated for all students. The entire population of students was divided into three groups: the *"⌐C ⌐Java"-group* (*mean score = 0.82±0.04*), the *"C ⌐Java"-group* (*mean score = 0.93±0.02*) and the "*Java"-group* (*0.98±0.01*). The data distribution in two groups was skewed from the normal and the variance of data across groups was not homogenous. Therefore, we used a non-parametric statistical test as a substitute for the one-way ANOVA. The Kruskal-Wallis test has demonstrated that there is a statistically significant difference in the cumulative level of Java knowledge among students from different groups: *Chi-Square statistics = 7.408; p = 0.006*.

Before saying that the automatic mapping of one model into another is possible and beneficial, it is worthwhile to validate the underlying assumption that the knowledge transfer from C to Java did happen. As a target set here we used students from both the *"⌐C ⌐Java"-group* and the *"C ⌐Java"-group*. We did not use the third group of students, since their performance on the Java pre-quiz depended directly on their previous Java knowledge, rather than on the knowledge of related C concepts. To estimate the strength of the relationship between the knowledge of the target sets of C and Java concepts, we applied a correlation analysis. The Pearson correlation could not be used, since our dataset did not meet a number of underlying statistical assumptions. However, the non-parametric Kendall's correlation analysis demonstrated a statistically significant relationship between the results on the C and Java pre-quizzes: *Kendall's Tau b = 0.294; p = 0.050*. Since students from the analyzed group did not have any previous knowledge of Java before the course, we can credit such a relationship to the transfer of knowledge from C to Java.

2.3. SM Translation Based on Automatic Ontology Mapping

To evaluate the quality of the SM translation based on automatic ontology mapping we compared it with "*ground truth*," as provided by the results of manual SM translation. We represented SMs as vectors, where every concept is a coordinate. As a result, the student knowledge of nine related C and Java concepts are characterized by two 9-dimensional vectors (as an example, fig. 3 shows the SM of knowledge of the C concepts *StatementFor* and *StatementIfElse* and the SM of knowledge of the corresponding Java concepts *for* and *else*, represented as 2-dimensional vectors). Since the manual mapping we performed was the best possible one-to-one mapping, we can say that the dimensions of the C and Java vectors are collinear. Hence, we can superimpose these coordinate spaces and calculate the Euclidian distance between two vectors, which reflects the divergence between the initial C and Java knowledge of the student, and, consequently, the divergence between the Manually-translated Java Model and the Original Java Model.

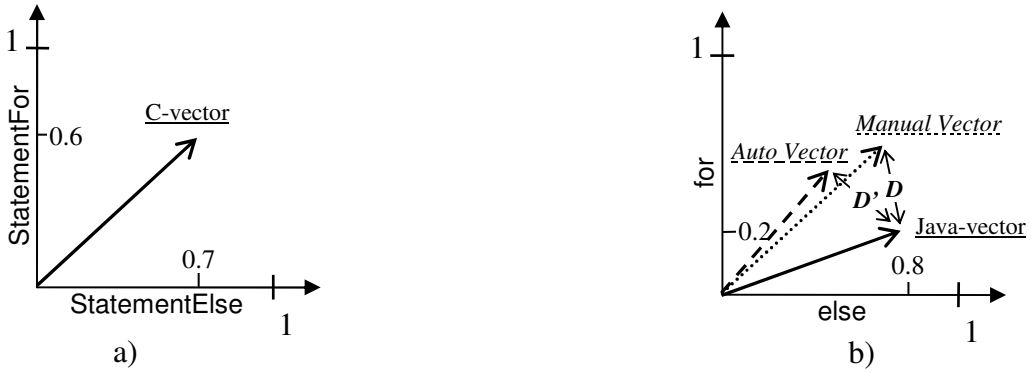

Figure 3: The SMs of knowledge for two concepts represented as 2-dimensional vectors.
a) The C-model (knowledge level of *StatementIfElse* is 0.7; knowledge level of *StatementFor* is 0.6).
b) The Java-model (knowledge level of *else* is 0.8; the knowledge level of *for* is 0.2) with the two other vectors superimposed: (i) Manually-translated Java vector (Corresponding concepts of the C and Java models are mapped to each other as 1-to-1; therefore the dimensions of the two coordinate spaces are collinear); (ii) Automatically-translated Java vector. Euclidian distance ***D*** shows the divergence between the Manually-translated Java Model and the Original Java Model, Euclidian distance ***D'*** shows the divergence between the Automatically-translated Java Model and the Original Java Model.

Based on the manual mapping as well as the results of the C pre-quiz and Java extra-credit questions, we calculated the divergence distances for all students from the experimental group (*"C⌐Java"-group*). These distances characterize both the shortcomings of the student modeling, due to the simplification of modeling assumptions and defects in elicitation process, as well as possible differences in the students' understanding of the relevant concepts of C and Java, unexpected mistakes (or guesses) on pre-quizzes, etc. Overall, these distances characterize the intrinsic imperfection of the obtained dataset. It is not possible to compute a set of smaller distances, i.e., to perform a better translation of the SMs on the basis of the given data.

Automatic mapping provided us with another set of distances. These distances characterize the differences between the original Java model vectors obtained from the Java pre-quiz and the Java model vectors computed by automatic translation of the SMs of C programming into SMs of Java programming. Every coordinate (every concept's level of knowledge) of later vectors is calculated based on the results of automatic ontology mapping and the knowledge of C concepts being mapped. To verify that the distances obtained manually are close enough to the estimates based on the automatic mapping, we applied correlation analysis. The results (*Kendall's Tau b = 0.844; p << 0.05*) demonstrate an exceptionally strong relationship between the two variables.

Hence, we have shown that the SMs of Java knowledge obtained with the help of automatic translation from the SMs of C knowledge are very close to those computed manually, i.e., automatic mapping provides close to "perfect" translation of knowledge levels from the C to Java domains.

Table 2. Results of the statistical analysis.

Goal	Type of analysis	Results
Validation of SM of C knowledge	Wilcoxon-Mann-Whitney test for 2 independent samples	Mann-Whitney U statistics = 19.0 (**p = 0.010**)
Validation of SM of Java knowledge	Kruskal-Wallis test for k independent samples	Chi-Square statistics = 7.408 (**p = 0.006**)
Relationship between C and Java knowledge	Kendall's test of correlation	Kendall's Tau b = 0.294 (**p = 0.050**)
Relationship between SM translations based on manual and automatic mapping	Kendall's test of correlation	Kendall's Tau b = 0.844 (**p << 0.05**)

3. Discussion and Future Work

The described experiment was performed for two related but different domains; however, the solution we propose could also be applied for translating SMs collected by different systems in the same domain. Two knowledge engineers would likely model the same domain in different ways. The modeling information is too precious; modern AESs need to be capable of translating each others' SMs.

On the other end of the scale lies the situation where the domains are fairly different, such as Biology and Geography. Can knowledge mediation in such domains still benefit by ontology mapping (probably for the very high-level concepts)?

A more theoretical question is: what are some general recommendations for close-domain ontology mapping? When is it worthwhile? What are the metrics of domain closeness? How can we say whether model translation is possible for two domains? It could be the percentage of related concepts. It could be the closeness of domains of interest in the general hierarchy of domains based on some common sense upper-level ontology, such as SUMO [22], DOLCHE [23] or CYC [24]. It could be some metric from the field of information retrieval, obtained by the analysis of related recourses.

The work described in the paper is only a first step. There are several possible directions of how we might continue this project:

- Differences in domain model representation are not the only source of discrepancy between overlay SMs. The scales used to assess knowledge levels should be mapped as well. They might be categorical vs. continuous; lower and upper borders could differ, as well as scale intervals. The formula for knowledge level calculation might also make a difference. For example, the same numeric value on the fixed scale calculated as a weighted average progress, or as a logarithm of the number of correct attempts, or as a sigmoid function value can represent different knowledge levels.
- Although the results of the mapping algorithm are satisfying, we could further improve its performance and precision. The current version on our server maps the C and Java ontologies used in about 30 minutes; such performance does not allow the building of an interactive application. It also needs to adjust to the possible extension of data sets and/or bigger ontologies. The utilization of structural relations between concepts as well as the development of a more elaborate, name-based mapping component should improve the precision of the algorithm. For example, we might use WordNet [25] for synonym processing or take into account the TFIDF values of different words in the concept names, in order to extract meaningful words from names like *StatementWhile, StatementIf, StatementFor*, etc.
- It is interesting to test the proposed approach in different domains. What would be the results for two different conceptualizations of the same domain? Or for domains less related then C and Java?
- Finally, it would be a good idea to develop an automated community service performing SM translation online.

4. Conclusions

In this paper, we investigated two main research questions:

- Is it possible to initialize a SM of an AES using another AES's SM acquired in a related but different domain?
- Is ontology mapping an appropriate mechanism for doing this?

The analysis of experimental data shows that automatic ontology mapping provides a sufficient basis for SM translation from one domain to another. Certainly, the obtained values are not perfect; however, they provide a better estimation of the starting levels of knowledge than the dominant *tabula-rasa* approach. The main advantages of the proposed approach are as follows:

- It is **fully automatic**: Once found, the mapping could be used for translating any number of SMs from one ontology to another.
- It is **bidirectional**: The found mapping could be used for the translation of SMs in both directions (for example, in the analyzed scenario: from both C to Java and Java to C).
- It is **instructionally-safe**: The translation process does not interfere with the learning process; students are not required to pass additional tests or to estimate their own knowledge.
- It is **objective**: Its performance is defined by the granularity and modeling constraints of the domain ontologies, as well as the accuracy of the mapping algorithm; it does not depend on any subjective decision made by a student or an instructor.
- It is **domain independent**: As long as two ontologies share some set of concepts, and mapping occurs, translation is possible.

We strongly believe that application of ontological technologies for student modeling can help to overcome a number of AES problems. The reported project is a step in this direction.

References

[1] Carbonell, J.R., AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction. IEEE Transactions on Man-Machine Systems, 1970. 11(4): p. 190-202.

[2] Brusilovsky, P. and C. Peylo, Adaptive and intelligent Web-based educational systems. International Journal of Artificial Intelligence in Education, 2003. 13(2-4): p. 159-172.

[3] Weber, G. and P. Brusilovsky, ELM-ART: An adaptive versatile system for Web-based instruction. International Journal of Artificial Intelligence in Education, 2001. 12(4): p. 351-384.

[4] Mabbott, A. & Bull, S., Student Preferences for Editing, Persuading and Negotiating the Open Learner Model, in M. Ikeda, K. Ashley & T-W. Chan (eds), Intelligent Tutoring Systems: 8th International Conference, Springer-Verlag, Berlin Heidelberg: p. 481-490.

[5] Berkovsky, S., T. Kuflik, and F. Ricci. Cross-Technique Mediation of User Models. in 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2006). 2006. Dublin, Ireland: Springer Verlag: p. 21-31.

[6] Carmagnola, F. and F. Cena. From Interoperable User Models to Interoperable User Modeling. in 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2006). 2006. Dublin, Ireland: Springer Verlag: p. 409-414.

[7] Melis, E., et al., Semantic-Aware Components and Services of ActiveMath. British Journal of Educational Technology, 2006. 37(3): p. 405-423.

[8] Kay, J., R.J. Kummerfeld, and P. Lauder. Personis: a server for user models. AH'2002, Adaptive Hypertext 2002. 2002: Springer: p. 203-212.

[9] Singley, M.K. and J.R. Anderson, The transfer of cognitive skill. 1989, Cambridge, MA: Harvard University Press.

[10] Carr, B. and I. Goldstein, Overlays: A Theory of Modelling for Computer Aided Instruction. 1977, Massachusetts Institute of Technology, Cambridge AI Lab. p. 24.

[11] W3C. Semantic Web. 2001. Available from: http://www.w3.org/2001/sw/.

[12] Kalfoglou, Y. and M. Schorelmmer, Ontology mapping: the state of the art. The Knowledge Engineering Review, 2003. 18(1): p. 1-31.

[13] Mizoguchi, R., Sinitsa, K. and Ikeda, M., Task Ontology Design for Intelligent Educational/Training Systems. Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs in conjunction with ITS'96, Montreal, 1996: p. 1-21.

[14] Mizoguchi, R. and Bourdeau, J., Using Ontological Engineering to Overcome AI-ED Problems, International Journal of Artificial Intelligence in Education, Vol.11, No.2, 2000: p.107-121.

[15] Denaux, R., Dimitrova, V., Aroyo L., Integrating Open User Modeling and Learning Content Management for the Semantic Web, User Modeling 2005: 10th International Conference, Edinburgh, Scotland, UK. Springer, 2005: p. 9-18.

[16] Trella, M., C. Carmona, and R. Conejo. MEDEA: an Open Service-Based Learning Platform for Developing Intelligent Educational Systems for the Web. Workshop on Adaptive Systems for Web-Based Education: tools and reusability (AIED'05). 2005. Amsterdam, the Netherlands: p. 27-34.

[17] Mitrovic, A., & Devedzic, V. (2004). A Model of Multitutor Ontology-based Learning Environments. Continuing Engineering Education and Life-Long Learning, 14(3), 229-245.

[18] Henze, N., P. Dolog, and W. Nejdl, Reasoning and Ontologies for Personalized e-Learning in the Semantic Web. Educational Technology & Society, 2004. 7(4): p. 82-97.

[19] Doan, A., et al., Learning to map between ontologies on the semantic web. 11th International World Wide Web Conference. 2002. New York, NY, USA: ACM Press: p. 662-673.

[20] Levenshtein, I.V., Binary Codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory, 1966. 10(8): p. 707-710.

[21] Maedche, A. and S. Staab. Measuring Similarity between Ontologies. 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web 2002: Springer-Verlag.

[22] Niles, I. and A. Pease. Towards a standard upper ontology. FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems. 2001. Ogunquit, Maine, USA: ACM Press.

[23] Gangemi, A., et al. Sweetening Ontologies with DOLCE. Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002. 2002. Siguenza, Spain: Springer Verlag: p. 166-181.

[24] Lenat, D.B., CYC: a large-scale investment in knowledge infrastructure. Communications of ACM, ACM Press, New York, NY, USA, 1995. 38(11): p. 33-38.

[25] Miller, G.A., et al., Introduction to WordNet: An On-line Lexical Database. International Journal of Lexicography, Oxford University Press 1990. 3(4): p. 235-244.