

# An Architecture for Combining Semantic Web Techniques with Intelligent Tutoring Systems

Pedro J. Muñoz Merino and Carlos Delgado Kloos

Carlos III University of Madrid, Department of Telematics Engineering,  
Avda de la Universidad, 30, E-28911 Leganés (Madrid), Spain  
{pedmume, cdk}@it.uc3m.es

**Abstract.** There are several possible applications of Semantic Web techniques in Intelligent Tutoring Systems (ITSs) and important advantages can be obtained from their application. This paper presents an architecture to combine Semantic Web techniques with ITSs, explaining its elements, relationships, challenges, and the different design criteria, offering some guidelines to make decisions when different implementation solutions are possible. We have implemented an instance of this architecture using the XTutor ITS and the CWM (Closed World Machine) Semantic Web reasoner. The implemented framework permits personalization of problems with hints for students. The paper also describes this specific implementation of the general architecture as well as some user adaptation examples with the implemented framework.

## 1 Introduction

There are many different ITSs; a categorization of them can be seen in [1]. The functions that such systems can perform vary; some examples are student assessment [2], or adaptive hypermedia [3], [4]. Some ITSs use educational information in a proprietary way, while others follow educational standards to enable interoperability.

There are several advantages in using Semantic Web techniques with ITSs, which justifies its combination. At present, few works have discussed architectures for enabling this combination. Reference [6] presents a Web service-based architecture to enable Semantic Web methods in adaptive hypermedia. Our architecture approach presents a different point of view, because it focuses on the relationship between Semantic Web reasoners (from now on reasoners) and existing ITSs, explaining its architectural elements, design criteria, implementation problems, etc. Moreover, a few works exist which explain specific system solutions but they do not cover the different architectural design criteria and implementation problems. In this work, we contribute to these issues, explaining an architecture for combining Semantic Web with ITSs that is general enough. Furthermore, we explain some challenges that Semantic Web techniques present when applied with ITSs; those challenges require architecture implementation decisions, and we give some recommendations.

We illustrate a specific case of the general architecture with the implementation of a framework to provide adaptive hints in problem-based learning based on this architecture and using the CWM reasoner [7], a new specification of hints we defined [8], and a hint player [8] that we implemented into the XTutor ITS [9].

The remainder of this paper is organized as follows. In Section 2, there is an outline of the related work about hint tutors and Semantic Web applications in education. Section 3 explains the general proposed architecture for combining Semantic Web with ITSs. Section 4 presents a specific implementation of the architecture for adaptive hints. In Section 5, some user adaptation examples with the implemented framework for adaptive hints are shown. Finally, Section 6 is devoted to the conclusions.

## 2 Related Work

### 2.1 Hint Tutors

It is clear that the provision of hints as a teaching methodology during problem solving has a positive impact on student learning (e.g. [10]). Therefore, several hint ITSs exist, as well as related works about this topic (e.g. [11], [12], [13]). Several of these systems allow the personalization of hints but using techniques different from the Semantic Web, such as Bayesian Networks (e.g. [11]).

### 2.2 Semantic Web Applications in Education

Ontology engineering is a key aspect for the success of Semantic Web. In [14] there is a clear vision of the use of ontologies for Artificial Intelligence in education. There are educational ontologies for different purposes such as competences [15], or domain ontologies that can be derived from the text [16]. A repository of ontologies for education was built [17]. Some ontologies have also been created to enable the inclusion of e-learning standards into the Semantic Web, such as for SCORM (Sharable Content Object Reference Model) [18], or IMS-LD (Learning Design) [19], [20].

There are different possible applications of Semantic Web in education. We focus on adaptive applications. The different types of adaptive hypermedia are explained in [3]. Adaptive applications using the Semantic Web have been proposed for contents [21], assessments [22] or educational feedback [23]. In the implemented framework described in this paper, we focus on a different domain, which is the provision of adaptive hints, taking into account the defined elements of our specification of hints.

## 3 General Architecture

Fig. 1 shows a graphical representation of the proposed architecture. Section 3.1 explains its elements and relationships. Section 3.2 describes the design criteria, analyzing the advantages and disadvantages of the different possible decisions.

### 3.1 Description of the Elements and Their Relationships

The *Data Storage* contains the main information that the system can use. The data is divided into two groups: *static* and *dynamic*. Information is considered static when it does not change (e.g. specific course objectives), while information is considered dynamic when it can change (e.g. the number of times a student has visited some content). In any case, teachers or system designers must provide all the static information at the beginning, and the initial state of dynamic information when applicable.

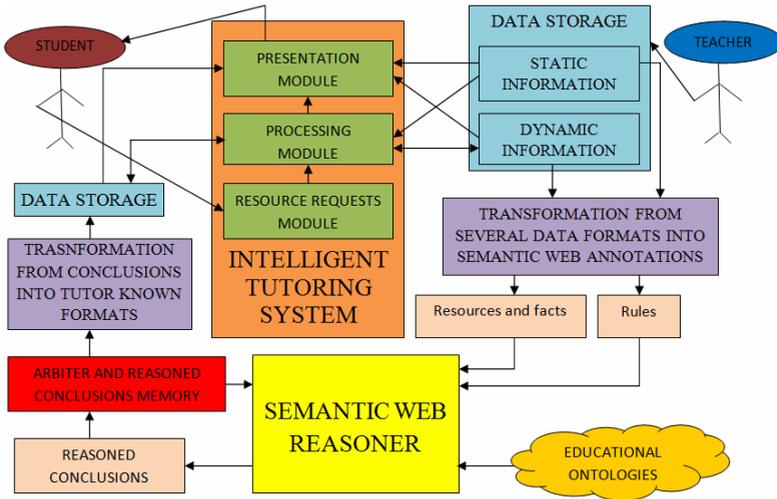


Fig. 1. General Architecture for Combining Semantic Web techniques with ITSs

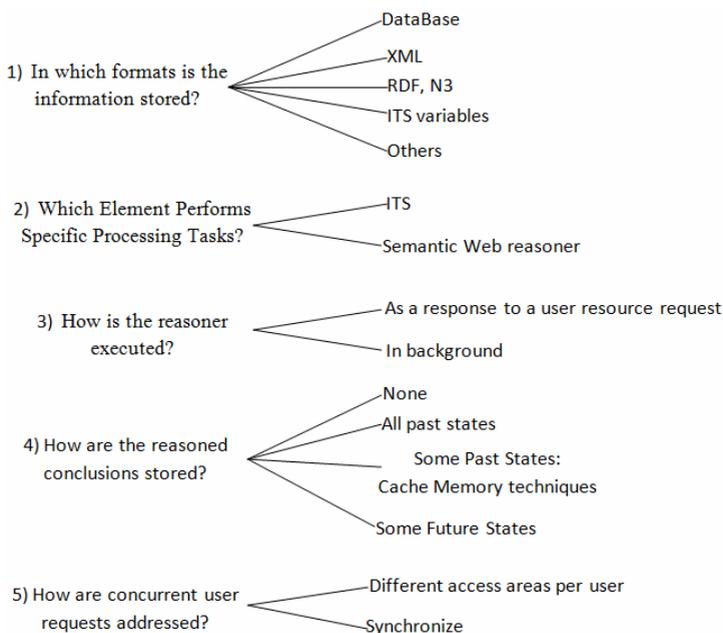
The *Semantic Web reasoner* is a tool that permits logic reasoning based on a set of *rules*, *resources* and *facts* according to defined *ontologies*, reaching *conclusions* as a result. Rules, resources and facts must be in specific formats (e.g. *RDF/XML* [24] or *N3* [25]). Therefore, a *transformation module* is required to convert the different information formats into Semantic Web annotations (unless some information resources are directly annotated in Semantic Web languages, for which the transformation module does nothing).

The *Arbiter and reasoned conclusions memory* selects the conclusions to store in each moment in order not to repeat already made inferences. The *reasoned conclusions* of the reasoner are in a format that is not usually understandable by ITSs. Therefore, another *transformation module* is required to convert those conclusions into ITS known formats. Here again, some conclusions may not need transformation if the ITS does not use this information (but the reasoner may use it again, if it represents feedback conclusions to the reasoner as shown in Fig. 1).

Lastly, the ITS receives *Resources Requests* from students that are then transmitted to the ITS *Processing module* which takes the proper actions, having data storage information as input. Dynamic data can be modified as a result of the ITS processing module. Finally, some resources (responses to requests) must be presented to the students, using the ITS *Presentation module* that can obtain any information from the Data Storage. Defined educational specifications may bring together (e.g. in XML files) processing and presentation information.

### 3.2 Design Criteria and Decisions

Several design criteria and decisions must be taken when implementing the architecture. Fig. 2 shows an overview of the different issues to decide.



**Fig. 2.** List of decisions to be taken in the defined architecture

**In Which Formats Is the Information Stored?** The same information can be stored in different formats (e.g. a data base format, XML, N3 or ITS variables). We must make a decision about which format to use, based on the following points:

- 1) *Desired level of interoperability.* The desired level of interoperability should be decided (e.g. with educational systems that interpret XML specifications, or with Semantic Web tools that interpret Semantic Web formats such as RDF or N3). It may be the case that interoperability is not important, so any formats are possible.
- 2) *Existing implemented tools:* For example, if we had an ITS player already implemented which interprets an XML format for assessments, then information about assessment description might be put into this format to take advantage of the tool.

Sometimes the information must be replicated in different formats. The designer can write the information directly or a transformation module can be used.

**Which Element Performs Specific Processing Tasks?** To decide which element will perform each specific task, we should achieve a balance between these issues:

- 1) *The formats of the existing information.*
- 2) *The desired rules for reuse.* If we want to reuse some Semantic rules between different systems (e.g. ITSs), then the reasoner should do the processing.
- 3) *Execution Time.* In general, an ITS performs a processing task more quickly than the reasoner. For tasks where execution time is critical, this is important.

**How is the Reasoner Executed?** The reasoner can be executed as a response to a student resource request or in background. The advantages of the former are:

1) *Responses to the students are up to date*, because the reasoner is executed for each specific request.

2) *It is not necessary to store past reasoned conclusions*. Since each request is transmitted to the reasoner, then the conclusion can be reached at that moment.

But the main advantage of executing the reasoner in background is the *reduction of the system response time to the user*. When a user request arrives, the system may respond to the user with some data that had been inferred previously, so there is no extra time for reasoning on that specific request.

Finally, it is important to note that both methods can be combined.

**How Are the Reasoned Conclusions Stored?** For each application, there is a set of different input combinations that need reasoning processing. For example, it could be the combinations between the different user states and educational resources. The different possibilities regarding which reasoned conclusions should be stored are:

1) *All past states*: All conclusions from any past states are stored. Indeed, all the states can be reasoned before the system starts working. With this solution, we save reasoning time, not having to repeat the same thing several times. This is recommended if there are few states, no storage limitations, and the response time is critical.

2) *Some past states*. Techniques similar to cache memory can be used to select the past stored states. This is recommended when there are a lot of different possible states, some memory limitation or the response time is not so critical.

3) *None*. This is recommended when the response time for a request is not critical.

4) *Some Future States*. It consists of doing the reasoning for selected future states considering the greatest likelihood of appearing based on the present ones. This is recommended in the same cases as *Some past states*, and they can be applied together.

**How Are Concurrent User Requests Addressed?** Concurrent user request problems occur whenever the reasoner or/and the ITS try to write simultaneously to data that is common to several users. Solutions can include a specific storage area for each user or synchronize the part of code that accesses shared resources. Another effect of the concurrency problem is that the response time can increase because of the techniques to avoid it. This is particularly important for the reasoner. For each user request, at least one process related to the reasoning is created. This is more time-consuming than a thread for each request. In addition, if there are synchronized parts, then only one request is at the CPU at each time which increments the response time.

## 4 Framework Implementation for Personalized Hints

This section presents a specific implementation of the general architecture for achieving personalized hints. Section 4.1 introduces XTutor and the extension module of hints we developed. Section 4.2 explains the specific elements of the architecture. Finally, Section 4.3 explains the different design criteria and implementation decisions.

#### 4.1 Overview of XTutor and Its Hint Player

XTutor [9] is an ITS developed at the Massachusetts Institute of Technology. It includes a library of standard tags to provide some types of questions for students. We complemented it with a new library (*hints.tags.py*) to allow XTutor to provide hints. Full details of our specification of hints are in [8]. The questions with hints are created as XML files (called XDOCs) and XTutor is able to interpret and run them [8].

#### 4.2 Elements of the Implemented Architecture

Fig. 3 shows the architecture that we implemented to obtain adaptive hints. This architecture is a particular case of the architecture presented in Fig. 1, and the same colour denotes the same function. In this implementation, the reasoner used is CWM [7], the ITS used is XTutor, and the Semantic Web language is N3 [25].

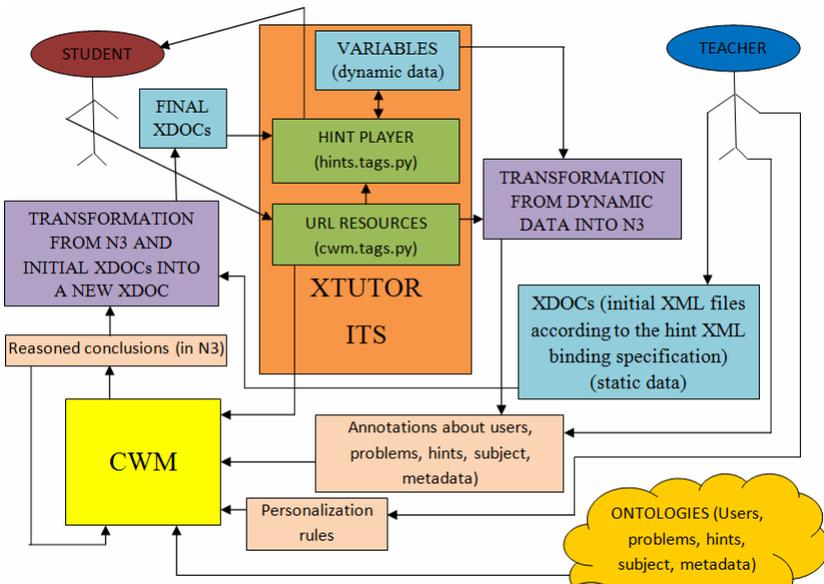


Fig. 3. Implemented Architecture for achieving personalized hints

The student requests an URL resource. This request is received by the *cwm.tags.py*. That is a program file we have implemented which executes some processing tasks. It calls CWM several times. Reasoning is performed by CWM based on the *personalization rules*, obtaining *conclusions*. The data that CWM receives as input are previous CWM conclusions related to the same user request; and *Semantic annotations* about users (preferences, knowledge, etc.), problems and hints (such as the number of students that solved the problem correctly without hints), and the subject domain. For each of these aspects, there is an *ontology*. Some static information (XML, rules and Semantic annotations) is written at the beginning (e.g. the *XDOC* initial files that teachers create that are XML files representing problems and hints), while other

information changes dynamically and is stored as XTutor *variables* (such as the number of students that answered a problem without hints correctly) and needs a conversion from XTutor to N3, so a *transformation module* is required.

Once the whole sequence of specific rules is performed, a final conclusion is obtained by CWM. This conclusion is related to the types, number of hints, etc. that will be applied to the requested resource. Next, as part of the *cwm.tags.py* file there is a *transformation module* from the N3 final conclusions and the initial XDOCs to a final XDOC compliant with the defined specification of hints. The initial XDOC is transformed to a different XDOC based on the reasoned conclusions. At present, the initial XDOC is not used as an input for reasoning by CWM so it is only generated in XML format by the teacher; however we will introduce it too in the future. To do so, a transformation module from XDOC formats into N3 annotations will be required.

Finally, the *cwm.tags.py* calls the hint player we implemented. This module receives the final XDOC as input and it runs it. The hint player performs processing and presentation tasks. The hint player processes information using state variables (e.g. present scoring, hints viewed or hints answered correctly), modifies variables according to the interaction, and presents an HTML response page to the student with the problem with hints. The problem with hints is personalized according to the conclusions inferred by CWM. Some of the variables modified in XTutor as a result of the interaction have an effect on the next CWM reasonings. For example, if a student responds incorrectly to a problem, then the student's knowledge level will decrease in the concepts covered by such problem. These necessary dynamic data variables are transmitted through the transformation module to obtain N3 annotations to be used in reasonings.

### 4.3 Decisions in the Specific Implemented Architecture

Now, we tackle the decisions made. Firstly, we have the following information:

- 1) *XDOCs describing problems with hints*. This is to allow interoperability at the XML level, to have data in a format understandable by the XTutor hint player (as it can perform quick processing and presentation of XML files), and because at this moment we do not need to reason about this aspect in CWM.
- 2) *Some information about users, problems, hints, subject concepts and rules in N3*. This is because we want to perform all the processing related to personalization with CWM (this is for code reuse, high level abstraction language, etc.), so it is necessary to write it in a CWM understandable format. In addition, we want to have Semantic Web interoperability of such aspects.
- 3) *Dynamic information that changes between interactions*. The database and the XTutor variables are the dynamic information. Since we do not need interoperability for this information, the quickest way for processing is to store such information in XTutor proprietary formats. But the information that is needed for reasoning is transformed, so this information will be replicated in two different formats.

CWM performs some processing tasks (to determine the personalized hints) because of rule reuse, while XTutor performs the other processing tasks.

At present, CWM is executed for each user request, but not in background. In addition, there are no conclusions stored (neither past nor future conclusions). This is

because we have not had time response problems with the present implemented rules and number of users accessing to the system. But in case rules implied a higher processing time, it would then be worthwhile to consider other techniques.

Finally, we are using different access areas per user to avoid concurrent user requests problems. There is no concurrency problem for static or dynamic data that is controlled by XTutor because XTutor controls concurrency by itself. The only data that can bring concurrency problems are the final XDOC generated. Note that if some conclusions were stored (past or future conclusions), then a possible solution would be to store them as a part of the final generated XDOC (this is permitted by the specification of hints defined). In this case the initial XDOC for each request would be the latest XDOC generated and it would be necessary to synchronize access to the final XDOCs, since several users may want to write concurrently in the same XDOC. Furthermore, CWM would be executed as a response to a request only in case the data related to the incoming request would not have been reasoned previously.

## 5 User Adaptation Examples

Figures. 4, 5, 6 and 7 show four different user adaptation examples in our framework. All the users request the same URL resource, so they obtain the same root problem (a multiple choice about servlets). But each student receives different personalized hints as a result of the reasoning performed by CWM. The first student (Fig. 4) does not receive a hint because he/she has a strong knowledge level in all the concepts.



Fig. 4. User 1 adaptation example

The second student (Fig. 5) receives one problem as a hint about the only concept included in the root problem that the student does not master. Among the candidate hints, the one whose level of difficulty is appropriate to the student level is selected.

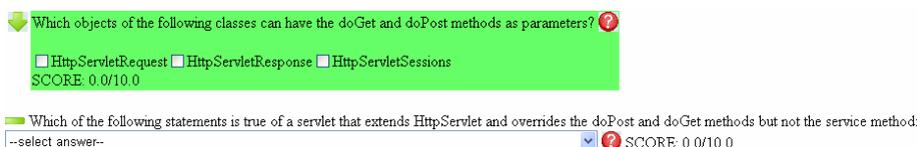


Fig. 5. User 2 adaptation example

Fig. 6 shows a student that has a low knowledge level for all the concepts included in the root problem, so he/she is presented with a sequence hint composed by four problems, one for each concept.

Which objects of the following classes can have the doGet and doPost methods as parameters? ?

HttpServletRequest  HttpServletResponse  HttpSession

SCORE: 0.0/10.0

Can you write an if within a servlet? --select answer-- ? SCORE: 0.0/10.0

By calling request.getParameter("parameter1") inside the doPost method you can obtain the value of parameter1, being parameter1 a form parameter with the value of the method attribute of the form written to Which value? ? SCORE: 0.0/10.0

You can execute a println method over an object of the class --select answer-- ? SCORE: 0.0/10.0

Which of the following statements is true of a servlet that extends HttpServlet and overrides the doPost and doGet methods but not the service method: --select answer-- ? SCORE: 0.0/10.0

Fig. 6. User 3 adaptation example

User 4 does not receive the hints immediately (Fig. 7). Instead, he/she receives a list with meta-information about the concepts covered for each hint he/she can see (which it is in our hint terminology a hint group). This is because User 4 has a combination of features in his/her personality model, so CWM always provides a hint group for this user, and it will always provide as many hint possibilities as concepts included in the initial root problem, but he/she will be able to select only a maximum of  $n$  hints, being  $n$  the number of concepts the student does not know well. In this case User 4 had a low level only on one concept, so he/she could only select a maximum of one hint out of the four available.

Each hint is about one of the possible initial choices  
You can select a maximum of 1 hint

- This hint is about the objects passed to doGet and doPost. You will loose 2 points for viewing it
- This hint is about conditional structures within servlets. You will loose 2 points for viewing this hint
- This hint is about the relationship between forms and servlets. You will loose 2 points for viewing this hint
- This hint is about servlets writing a response. You will loose 2 points for viewing this hint

Which of the following statements is true of a servlet that extends HttpServlet and overrides the doPost and doGet methods but not the service method: --select answer-- ? SCORE: 0.0/10.0

Fig. 7. User 4 adaptation example

In these user adaptation examples, the number and types of hints provided, the concepts chosen, etc. for each specific student are a result of the reasoning by CWM based on the different data. Finally, the conclusion results are transformed into XML files understandable by the XTutor hint module, which is the one that performs the final presentation to the users.

## 6 Conclusions and Future Work

In this paper we have presented an architecture for combining Semantic Web techniques with ITSs. This architecture is feasible to develop as we have implemented a specific case of it, for the provision of personalizing hints using the CWM Semantic Web reasoner, and the XTutor ITS.

During the implementation we encountered different problems and specific design criterions were required. We analyzed these different problems and made decisions for our particular case, but then we generalized the different design criterions which we explain in this paper together with some guidelines for taking decisions.

The implemented framework introduces Semantic Web techniques in the provision of adaptive hints that are personalized for students. In addition, it combines the use of a new defined XML specification of hints, the XTutor hint player that we implemented, the CWM reasoning capabilities, and other features of the XTutor.

At present, we are working in introducing more personalized rules in the system, extending the existing ones. We are planning to introduce this framework in classroom during this year.

**Acknowledgments.** Work partially funded by Programa Nacional de Tecnologías de la Sociedad de la Información, MEC-CICYT project MOSAIC-LEARNING TSI2005-08225-C07-01 and 02.

## References

1. Murray, T.: Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education* 10, 98–129 (1999)
2. VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L.: Andes physics tutoring system: Five years of evaluations. In: *12th International Conference on Artificial Intelligence in Education*, pp. 678–685. IOS Press, Amsterdam (2005)
3. Brusilovsky, P.: Adaptive Hypermedia. *User Modeling and User-Adapted Interaction* 11(1/2), 87–110 (2001)
4. Brusilovsky, P., Peylo, C.: Adaptive and intelligent Web-based educational systems. *International Journal of Artificial Intelligence in Education* 13(2-4), 159–172 (2003)
5. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American* (May 17, 2001)
6. Henze, N., Krause, D.: Personalized access to web services in the semantic web. In: *The 3rd Int. Semantic Web User Interaction Workshop, SWUI* (2006)
7. Berners-Lee, T.: CWM, <http://www.w3.org/2000/10/swap/doc/cwm>
8. Muñoz Merino, P., Delgado Kloos, C.: A software player for providing hints in problem-based learning according to a new specification. *Computer Applications in Engineering Education* (Accepted January 12, 2008) (in Press, 2008)
9. XTutor Intelligent Tutoring System, <http://xtutor.org/>
10. Harskamp, E., Ding, E.: Structured collaboration versus individual learning in solving physics problems. *International Journal of Science Education* 28(14), 1669–1688 (2006)
11. Gertner, A., Conati, C., VanLehn, K.: Procedural Help in Andes: Generating Hints using a Bayesian Network Student Model. In: *15th National Conference on Artificial Intelligence*, Madison, pp. 106–111 (1998)
12. Guzman, E., Conejo, R.: Self-assessment in a feasible, adaptive web-based testing system. *IEEE Transactions on Education* 48(4), 688–695 (2005)
13. Razaq, L., Heffernan, N.: Scaffolding vs. hints in the Assistment System. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) *ITS 2006*. LNCS, vol. 4053, pp. 635–644. Springer, Heidelberg (2006)

14. Riichiro, M., Bourdeau, J.: Using Ontological Engineering to Overcome Common AI-ED Problems. *International Journal of Artificial Intelligence in Education* 11, 107–121 (2000)
15. Lefebvre, B., Gauthier, G., Tadié, S., Duc, T., Achaba, H.: Competence ontology for domain knowledge dissemination and retrieval. *Applied Artificial Intelligence* 19(9), 845–859 (2005)
16. Zouaq, A., Nkambou, R., Frasson, C.: Building Domain Ontologies from Text for Educational Purposes. In: Duval, E., Klamma, R., Wolpers, M. (eds.) *EC-TEL 2007*. LNCS, vol. 4753, pp. 393–407. Springer, Heidelberg (2007)
17. Dicheva, D., Sosnovsky, S., Gavriloova, T., Brusilovsky, P.: Ontological Web Portal for Educational Ontologies. In: *Workshop on Applications of Semantic Web in E-Learning at 12th International Conference on Artificial Intelligence in Education*, Amsterdam (2005)
18. Aroyo, L., Pokraev, S., Brussee, R.: Preparing SCORM for the Semantic Web. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) *OTM 2003*. LNCS, vol. 2888, pp. 621–628. Springer, Heidelberg (2003)
19. Psyche, V., Bourdeau, J., Nkambou, R., Mizoguchi, R.: Making Learning Design Standards Work with an Ontology of Educational Theories. In: *12th Artificial Intelligence in Education*, Amsterdam, pp. 539–546 (2005)
20. Lama, M., Sánchez, E., Amorim, R.R., Vila, X.A.: Semantic Description of the IMS Learning Design Specification. In: *Workshop on Applications of Semantic Web in E-Learning*, Amsterdam, pp. 37–46 (2005)
21. Henze, N., Dolog, P., Nejd, W.: Reasoning and Ontologies for Personalized E-Learning in the Semantic Web. *Journal of Educational Technology and Society* 7(4), 82–97 (2004)
22. Cheniti-Belcadhi, L., Henze, N., Braham, R.: An Assessment Framework for eLearning in the Semantic Web. In: *12th GI- Workshop on Adaptation and User Modeling in interactive Systems*, Berlin, pp. 11–16 (2004)
23. Jovanović, J., Gašević, D., Brooks, C., Eap, T., Devedzic, V., Hatala, M., Richards, G.: Leveraging the Semantic Web for Providing Educational Feedback. In: *7th IEEE ICALT Conf.*, Niigata, pp. 551–555 (2007)
24. RDF/XML Syntax Specification,  
<http://www.w3.org/TR/rdf-syntax-grammar/>
25. Berners-Lee, T. (ed.): Notation3,  
<http://www.w3.org/DesignIssues/Notation3.html>