L. Aroyo, D. Dicheva & A. Cristea, Ontological Support for Web Courseware Authoring, ITSí02, Intelligent Tutoring Systems, LNCS 2363, Springer, 270-280

Ontological Support for Web Courseware Authoring

Lora Aroyo[1], Darina Dicheva[2] and Alexandra Cristea[1]

[1] Technische Universiteit Eindhoven, The Netherlands
{l.m.aroyo, a.i.cristea}@tue.nl
[2] Winston-Salem State University, United States of America
dichevad@cs.wssu.edu

Abstract. In this paper we present an *ontology-oriented authoring support* system for Web-based courseware. This is an elaboration of our approach to knowledge classification and indexing in the previously developed system AIMS (Agent-based Information Management System) aimed at supporting students while completing learning tasks in a Web-based learning/training environment. By introducing *ontology-based layers* in the courseware authoring architecture we aim at using *subject domain ontology* as a basis for formal semantics and reasoning support in performing generic authoring tasks. We also focus on *cooperative authoring*, which allows re-usage and sets the basis for *authoring collaboration*. To exemplify our method we define a set of generic tasks related to *concept-based courseware authoring* and present their ontological support by the newly added operational and assistant layers in the AIMS architecture.

1 Introduction

Courseware and its authoring acquire a new meaning in the context of Web-based education. Courseware, a term initially coined to name computer-supported presentation and use of teaching material aimed at improving the studentís course work by instruction individualization, traditionally consists of *teaching material*, divided into learning units (or frames), and a *courseware delivery engine*. The goal of courseware authoring was to support authors in creating and linking frames. The second generation was the *multimedia courseware* [4,10], based on the same principles but allowing multi-modality in material presentation thus significantly improving content presentation. The third generation was the *hypermedia courseware* [6,9]. The novelty was to remove the constraints of predefined paths in the learning material. In adaptive *educational hypermedia* [7,11] a student could browse and explore e-book links and pages by following an adaptable knowledge path. However, frames were still locally stored documents.

Nowadays, with the revolution that the Web has brought to information access worldwide, the meaning of courseware is changing again. Web-based courseware can be viewed as a *gateway* to a variety of Web educational materials related to specific topics or educational goals, developed by the course author (instructor) and stored

locally, or represented by Web addresses and descriptions. This dramatic change obviously affects the courseware authoring process, too [1,18, 19].

To efficiently organize and maintain Web-based resources we employ a powerful approach for *knowledge classification and indexing* in on-line learning environments, based on conceptualisation of the course subject domain. A significant aspect of the proposed approach is building a *subject domain ontology* [2], in line with recent Semantic Web research (e.g., layered architecture [5]) and ontologies [12].

In this paper we present our view on ontological support of Web-based courseware authoring, which is an elaboration of our approach to knowledge classification and indexing, aimed at supporting students in retrieving, evaluating, and comprehending information when performing learning tasks in a Web-based learning/training environment. We start by shortly presenting authoring Web-based courseware with AIMS. Next we propose a layered approach to support courseware authoring. Then, we define a set of generic tasks related to concept-based courseware authoring and their possible support. Finally, we present some conclusions and future perspectives.

2 AIMS ñ an Example of Courseware Authoring

From an enterprise point of view, the main actors in the proposed solution for information handling in a Web-based educational system (e.g., AIMS), are *student*, *courseware author* and *administrator* (with their set of responsibilities and tasks). The student is responsible for interpreting and processing information. Authoring includes information maintenance, i.e. creating, editing, structuring. The administrator is responsible, among others, for system information-access policies. In this paper we focus only on authoring roles related to domain and course authoring. By supporting these activities we aim at increasing the efficiency of information reuse and of collaboration between course authors. The AIMS authoring environment consists of three main modules: Domain Editor, Library Editor, and Course Editor [3]. These three modules correspond to the three layers in the systemís information base: library metadata, domain ontology, and course information (Fig. 1). The last two are represented as concept maps (CM) of domain concepts and links among them.

The *Domain Editor* enables the author to construct a domain concept mapping structure. It provides facilities to add, delete and update domain terms and links between them. For each new term the author specifies a name and definition along with its classification in a simple hierarchy within the concept map (including category, sub-category, topic and sub-topic). The editor also allows authors to create new types of links and links between a domain term and existing documents in the AIMS library.

The *Course Editor* enables the author to define a structure of a course within a specific domain by using domain terms as basic framework of the structure definition. It allows the author to define course topics and course tasks and relate them to domain terms by assigning a list of keywords to each task. Each topic and task is given a definition and a reference (link to a main document in the library). One course can consist of several topics and each topic can have several tasks. A topic usually corresponds to the course weekly session and the tasks to the course weekly assignments. As the tasks

are directly related to domain terms and the domain terms to library documents, this provides a link between the course structure and the appropriate course material.

The *Library Editor* provides means for maintaining a collection of information related to different courses and domains. It provides simple options typical for most of the library systems. Each document is described both task- and use-oriented. For example, each document description includes its instructional and presentation formats, indicating the way this document could be used for instructional purposes and whether it is in an appropriate presentation format. Each document description includes also a list of keywords (not necessarily belonging to the domain ontology).

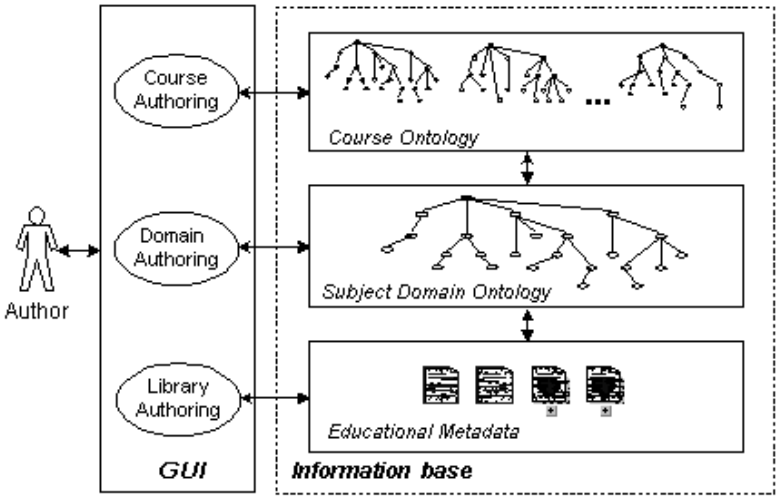

Fig. 1. AIMS authoring architecture

Since the authoring of concept-based Web courseware is three-fold, including domain-, course-, and library authoring, this process is more complicated and labor intensive than the process of ëstandardí courseware authoring. Such authoring is extremely difficult and time-consuming and needs specialized, modern authoring support and re-usage [2,17], cooperation and collaboration among authors. The further needs of support in concept-based courseware authoring are the following:

- automatic or semi-automatic performance of some authoring activities,
- intelligent assistance to the author in the form of hints, recommendations, etc.,
- supporting the activities of different instructors for collaborative building and/or cooperative reuse of domain and course ontologies.

Collaborative authoring [19] occurs in project-like settings, where authoring subtasks are delegated to a group of authors. This kind of authoring needs synchronization, dialogue support and coordination of the whole project. In contrast, *cooperative authoring* [8] mainly involves asynchronous re-usage of authoring products, such as course materials, libraries, ontologies, etc. In our current work we focus on the latter, more precisely, on supporting primitive interaction activities in cooperative authoring.

The key idea of our approach is that to provide enhanced support for authoring concept-based Web courseware we use the systemís domain concept map, i.e. the ontology, which captures the semantics of the subject domain terminology used by students when searching for relevant information necessary to perform course tasks. The same ontology can be used by courseware authors to ask authoring-related questions or by the system to perform (semi-)automatically some authoring activities. Thus we introduce additional *ontology-based layers* to the courseware authoring architecture, which allow *intelligent assistance* of courseware authors. The idea is to use the existing domain and course ontologies for all generic authoring tasks related to concept-based Web courseware authoring, as a basis for formal semantics and reasoning support. In our work the ontologies are represented as concept maps [3,8]. Consequently, authoring involves manipulating concept maps, i.e., creating and modifying CMs. The proposed semantic layers for intelligent authoring assistance include reasoning, consistency check, and introduce additional operations on CMs (such as comparing CMs, mapping and merging CMs, extracting subsets of CM, analyzing CMs). Next we describe the suggested layered approach to support Web courseware authoring.

3 Ontology-based Layered Support to Courseware Authoring

To support cooperative concept-based courseware authoring, we aim at creating a re-usage based cooperative environment [19]. The primitive interaction activities among participants in this environment during both cooperative and collaborative authoring are: planning/creation, data/idea sharing, coordination/control/initiative/supervision, observation/suggesting and dialogue (with interaction). The support system should provide all appropriate tools for these activities. Furthermore, refined cognitive tools (e.g., concept mapping tools [1,8]) are required to facilitate group collaborative authoring [19], corresponding to the activities enumerated above. We propose a 2D-layer approach (Fig. 2) for concept-based courseware authoring support. This approach allows re-usage, in the sense of authoring cooperation, and sets the basis for authoring collaboration. The *Y-axis* represents the main information objects in the information base of the courseware system (library objects, domains, courses). The *X-axis* targets systemís support for information objects authoring (*GUI, Assisting -, Operation -, Information layer*) and is represented by a layered architecture implementing system functionality. The *GUI layer* supports user-system communication. The *Information layer* contains the layered description and structuring of the information objects in the courseware system (*educational metadata, subject domain ontology, course ontology*). The educational metadata layer contains the description of the data sources. The two new layers in the extended architecture are the *Assisting* and the *Operation layer*.

The *Operation layer* handles the operations related to data in each information layer thus providing means for modeling data into ontology and creating alternative goal-oriented structures of courses. The Operation layer is also responsible for facilitating information manipulation, consistency and co-operation. It consists of three

processing engines: (a) *course engine*, (b) *domain engine* and (c) *library engine*. All of them include two types of support operations: (a) *consistency check*, and (b) *co-operation support*. The consistency modules perform their activities over each sub-layer within the information layer. They provide functions to facilitate the process of authoring the domain ontology, course ontology and educational metadata in a semi-automatic way. These modules also guarantee the consistency of the educational sources [17]. They should deal with tasks such as: handling notions of *semantic equivalence* [21] and conflict, conflict resolution rules, equivalence comparison rules, enhancing the resulting ontology and defining additional constraints if necessary.

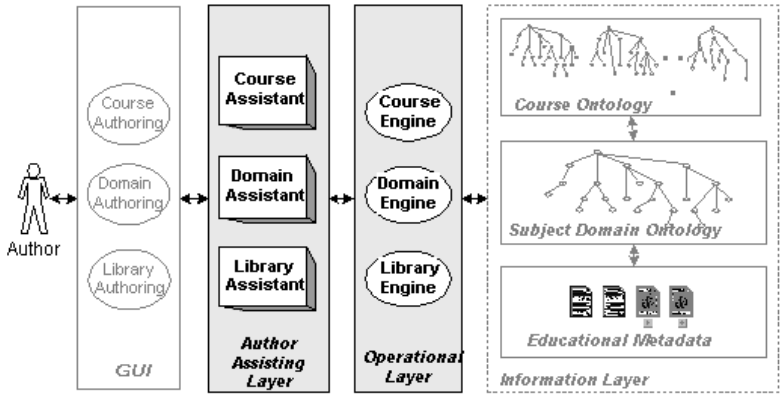

Fig. 2. 2D-layer approach towards courseware authoring support

The co-operation support modules offer on one hand a set of operations to check the consistency in alternative (simultaneous) course structure building by different authors and on another - predefined functions (patterns and templates) to facilitate effective reusability of the available course structures developed by different authors.

Concerning the reusability support, we pay special attention to the issues associated with merging ontologies [16], such as: extracting portions of an ontology to be merged with another [21], identifying which frames are to be extracted from the source ontology, determining if the extracted information has semantic overlaps or conflicts with the target ontology, assisting in merging ontologies, recording the sources of inserted sub-ontologies for later reference and update, selecting patterns, templates in an educational ontology for presenting them as predefined objects to other authors.

Among the issues of importance when merging two ontologies are those related to the following types of semantic overlaps and conflicts: semantically equivalent concepts but with different names [21], semantically different concepts but with the same name, semantically equivalent concepts with the same name but different definitions, semantically equivalent concepts linked to different /conflicting concepts, etc.

While the Operation layer actually implements the authoring operations, the Assisting layer, which is based on the ontological mapping of the domain, is responsible for helping the author in the process of courseware authoring. For example, it gives hints

to the author of how to create a course structure, or how to link a document to the ontology, or how to link a course item to the ontology, etc.

According to the computational semantics of an ontology [15], the ontologies we consider here can be situated at *level 1* (term collection, as shown previously (Fig. 1,2) and *level 3* (executable task ontologies). We still lack the connection given by *level 2* (formal definitions, constrains and axioms).

4 Generic Authoring Tasks Support

In this section we discuss generic authoring tasks supported by the operation sets [15] of the Operation layer and the presentation options provided by the Assisting layer.

Table 1 Atomic operation definitions

Atomic operation	**Range**	**Description**
ëAddí	performed over sets of objects $\{T_o, T_a, C_o, L_i, D_{oc}\}$, where: $T_o \in$ *{course topics}*, $T_a \in$ *{course tasks}*, $C_o \in$ *{domain concepts}*, $L_i \in$ *{domain links}*, $D_{oc} \in$ *{library documents}*.	adds each object to either course structure, domain ontology or metadata library.
ëDelí,	as above	deletes an object from the corresponding structure
ëEdití	as above	edits the object settings
ëUí	set *{CM, CS, EML}*, where: *CM*=Concept Map, *CS*=Course structure, *EML*=Educational Metadata Library.	ensures current state update of the corresponding information structure of set
ëLí	sets $\{DirLC, RelC, RelC_o, RelT_a, RelD_{oc}\}$, where: $DirLC_o$ = Directly linked concepts, *RelC* = Related courses, $RelC_o$ = Related concepts, $RelT_a$ = Related tasks, $RelD_{oc}$ = Related documents.	lists the objects of the set(s)
ëVí	set *{Graph, Text}*, where ë*Graphí* is a graphical and ë*Textí* gives a textual results view.	gives alternative views of the *engine* results to the author
ëChkí	set $\{T_a, T_o, C_o, L_i, D_{oc}, RelC_o, RelT_a, RelD_{oc}, DirLC\}$	checks the existence of objects within the set(s)

We are defining a complete set of generic authoring tasks at all three information layers (course, subject domain and library) that are supported by the course, domain and library engines. In this paper however we present only an excerpt from the course engine supported authoring tasks (Table 2). We further illustrate the interaction between the course engine and the course assistant in supporting the author by presenting an activity diagram of the support for the atomic authoring task ëadd topicí (Fig. 3). Table 1 above presents abbreviations and definitions of atomic operations used in this section. There are number of composite actions such as ëdelete all topics of a courseí, ëdelete all concepts of a topicí, ëdelete all tasks of a topicí, ëdelete all concepts of a taskí or ëgive value ëaí to all the concept weights of a taskí, which can be implemented with a repetitive call to the atomic operations called ëdelete topicí and ëlist all topicsí and the corresponding operations for tasks and concepts. In Table 2 we present an excerpt of the course authoring ontology with a selection of basic atomic tasks and the interaction between course engine and assistant.

Table 2 Course engine supported authoring tasks

Task	Course Assistant	Course Engine	Result
Add (T_o, CS)	▪ suggest options for the author: ▪ add or delete *course engine* results ▪ give alternative presentation: – V (Text, RelC, Relevance %) – V (Graph, Course Trees, Matched Concepts) ñhighlighted – V (Graph, Domain Ontology, Matched concepts) ñ ëyou are hereí ▪ notify other authors of adding T_o to CS	▪ perform a keyword search on T_o expression within: - DO (domain &) - CO (course ontology) ▪ store results for reuse ▪ U (T_o) ▪ U (CS)	▪ L (RelC, keyword percentage) ▪ L ($RelC_o$, depth within DO)
Add (T_a ,T_o)	▪ if a new connection to the domain concept is discovered, options are: – Del (connection) – Add (C_o, T_o) – Add (C_o, T_a) – V ($RelT_a$) – Copy ($RelT_a$) ▪ notify other authors of adding T_a to T_o	▪ Chk (T_a, ∃) = true ▪ Chk (T_o, C_o, compatibility) = true ▪ deduce *assistant* activity ▪ Chk ($RelT_a$, other courses) = true ▪ U (T_o) U (CS)	▪ L ($RelT_a$, other courses) ñ ordered by their weight-related relevance ▪ L ($RelT_a$, same course)
Add (C_o ,T_a)	▪ if Chk (C_o, T_a, ∃) = true: – Notify the user – Change the weight of the C_o ▪ if Chk (C_o, T_a, ∃) = false →V (C_o, T_a) ▪ notify other authors of adding C_o to T_a	▪ Chk (C_o, T_a, ∃) = true ▪ U (T_a, C_o) ▪ U (T_o) ▪ U (CS)	▪ L (T_a, ɜ C_o) ▪ L (T_a, other courses) ▪ L ($RelC_o$) ▪ L (all C_o, T_a)
Add (C_o ,T_o)	▪ if Chk (C_o, T_o, ∃) = true ▪ notify the user ▪ if Chk (C_o, T_o, ∃) = false ▪ V (C_o, T_o) ▪ notify other authors of adding C_o to T_o	▪ Chk (C_o, T_o, ∃) = true ▪ U (T_o, C_o) ▪ U (CS)	▪ L (T_o, ɜ C_o) ▪ L (T_o, other courses) ▪ L ($RelC_o$) ▪ L (all C_o, T_o) ▪ L (T_a, ɜ C_o) ▪ L (T_a, other courses)
E (C_o T_a)	▪ V (options to choose): – change (C_o, weight) – Del (C_o, T_a) – Del (C_o, $RelT_o$) ▪ Notify other authors of editing T_a in C_o	▪ U (C_o W) ▪ U (C_o) ▪ U (C_o T_a) within different system modules	L (C_o T_a)
Del (C_o ,T_o)	▪ if Chk (C_o, T_a, ∃) = true ▪ notify the user ▪ V (options to choose): – Del (Co, all Ta) – Del (Co, some Ta) – Del (Co, To) – Cancel Del option ▪ notify others of deleting T_o in C_o	▪ Chk (C_o, T_a, ∃) = true ▪ Chk (C_o, T_o, ∃) = true ▪ U (T_o) within different system modules ▪ U (CS)	L ($RelC_o$, T_o) - updated

Due to space restrictions we are not presenting the full ontology for the domain and educational metadata authoring. Other possible course authoring tasks not mentioned here relate to document library and education metadata: ëlink a document to a topicí, ëlink a document to a taskí, ëdelete a document from a taskí and ëdelete a document from a topicí.

Domain. The interaction stream between *domain assistant* and *engine* is triggered by common authoring tasks, e.g., ëcreate/edit/copy-domainí,ëmerge-domainsí. These tasks involve basic concept-maintenance such as ëadd/delete/edit conceptí, ëcreate/delete/edit link/typeí between concepts. At a higher level, authoring tasks include:ëremove-all-direct-links-to-conceptí,ëremove-all-segments-of-a-path-between-

two-conceptsí, ëedit/create-the-domain-mapí (ontological domain structure) or ëmake-links-between-domain-structure-and-libraryí. These tasks trigger operations performed by the domain engine over the ontological domain structure.

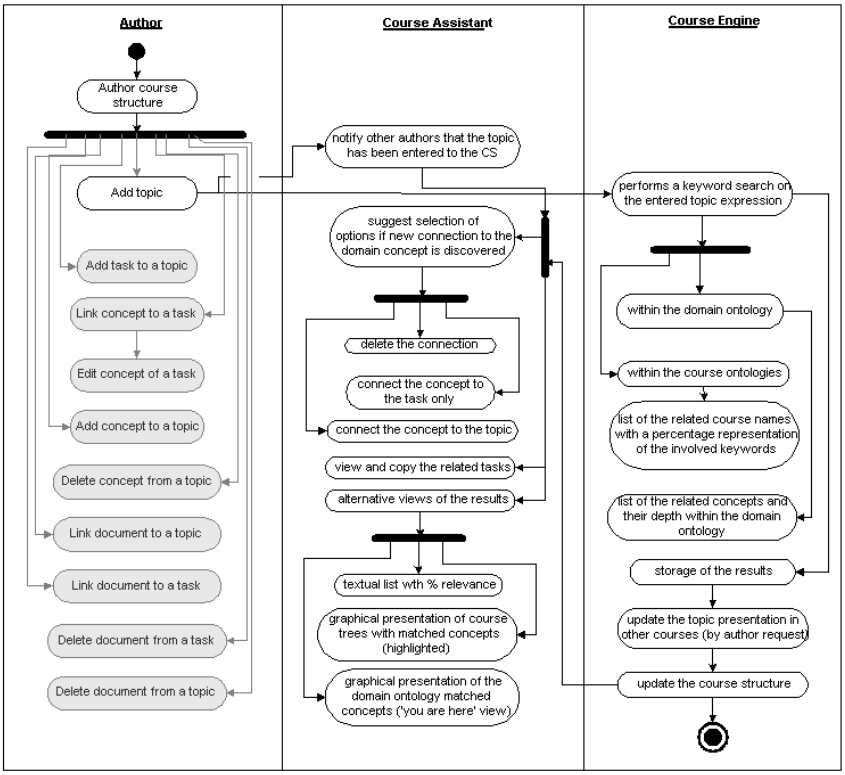

Fig. 3. UML activity diagram for the atomic authoring task ëadd topicí: *Add(T_o, CS)*

The operations ensure data consistency by performing domain specific checks for conflicts. For instance, when the authoring task *Add(C_o, CM)* is performed by the author (Fig. 3, Table 2 task 1) the domain engine performs *Chk(C_o, CM, exist)*, i.e. checks whether the concept Co is already in the map, updates the CM with *U(C_o, CM)*, performs *Add(C_o, weight)* and finally provides the results to the domain assistant for analysis and presentation to the author. Depending on whether the concept has been found in the CM, the domain engine returns: (a) *L(C_o, synonyms)* (b) *L(DirLC_o)* and (c) notification that the new concept C_o has been added to the CM. These results are input to the domain assistant, which is responsible for the customization and presenting them in an appropriate format to the author so as to support his/her task most efficiently. In this case the domain assistant performs the alternative operations allowing the author to choose from *V(Text, DirLC_o), V(Graph, DirLC_o)* and another set of alternative views for the synonyms *V(Text, C_o, synonyms), V(Graph, C_o, synonyms)*. There

are a number of composite actions such as ëdelete all direct links of a given conceptí or ëdelete all segments of a path between two conceptsí, which can be implemented with a repetitive call to the atomic operation called ëremove a link in the CMí.

Library. The interaction stream between the library assistant and library engine is triggered by a set of common authoring tasks, such as ëcreate/edit existing libraryí, ëadd/delete documentí, ëlink/unlink a document to domain conceptsí, ëadd/delete keywords to a documentí, ëedit the weights of the keywordsí, ëlink/unlink a document to course topics and tasksí. Due to lack of space the details of all possible library-authoring tasks are skipped.

5 Conclusions

In this paper we have introduced a 2D-layer approach to support Web-based courseware authoring. The main idea is to use *systemís domain ontology*, capturing the semantics of the subject domain terminology, in order to provide enhanced authoring support for concept-based courseware. We propose introducing additional ontology-based layers to the courseware authoring architecture, which allow intelligent authoring assistance. We elaborate on the various types of support that these layers should provide for the authoring actions within a courseware-authoring environment (for example AIMS, but not restricted to it). We consider also issues of re-usage and co-operative information sharing, towards collaborative authoring, in the sense of simultaneous performance of authoring activities. This is motivated by the increase in need for authorsí cooperation and collaboration, especially in Web-authoring, where information is plentiful and has only to be molded into the different shapes adequate for learning.

The processing presented in the paper is self-contained. However, many more aspects can be analysed, and further research direction pursued. A direction already pointed to in section 2 is towards collaborative authoring environments. This would mean a merge between re-usage based cooperative environments, such as the one presented here, and collaborative means of working extracted from previous researches on collaborative learning environments. Furthermore, such environments can benefit from the creation of a user model of the author. Another important direction is towards merging of ontologies. Here we will rely heavily on the developments and research in this field [21].

This paper represents a contribution towards collaborative and cooperative courseware authoring by both structuring, and adding semantics to the courseware in the sense of the standardization efforts of the semantic Web community.

References

1. Aroyo (2001). Task-oriented approach to information handling support within Web-based education. PhD Thesis, University of Twente, The Netherlands

2. Aroyo L., Dicheva D. (2001). AIMS: Learning and Teaching Support for WWW-based Education. Int. J. for Continuing Eng. Education and Life-long Learning, 11(1/2), 152-164.
3. Aroyo L., Dicheva D., Velev I. (2001). A Concept-Based Approach to Support Learning in a Web-based Course Environment. J. Moore et al. (Eds.) AI in Education, Amsterdam: IOS Press, Frontiers of AI and Applications, 68, 1-10.
4. Benyon, D., Stone, D., & Woodroffe, M. (1997). Experience with developing multimedia courseware for the World Wide Web: the need for better tools and clear pedagogy, Int. J. HumanñComputer Studies, 47, 197ñ218
5. Berners-Lee, T. (1998). Semantic Web road map. Internal note, World Wide Web Consortium. http://www.w3.org/DesignIssues/Semantic.html.
6. Bolter, J.D., Joyce, M., Smith, J.B. (1990) Storyspace: Hypertext Writing Environment for the Macintosh. Computer Software. Cambridge, MA: Eastgate Systems.
7. Brusilovsky, P. (2001). Adaptive Educational Hypermedia. In: Proceedings of Tenth International PEG conference, Finland, pp. 8-12.
8. Cristea, A. and Okamoto, T., (2001). Object-oriented Collaborative Course Authoring Environment supported by Concept Mapping in MyEnglishTeacher, Educational Technology and Society, 4 (2), April, http://ifets.ieee.org/periodical/vol_2_2001/v_2_2001.html
9. De Bra, P. & Calvi, L. (1998). AHA! An open Adaptive Hypermedia Architecture, The New Review of Hypermedia and Multimedia, 4, 115-139.
10. De Vries, E. (1996). Educational multimedia for learning and problem solving, EuroAIED: European conference on artificial Intelligence in Education, Lisbon, 157-163.
11. Kay, J. & Kummerfeld,B (1994). Adaptive Hypertext for Individualised Instruction, Workshop on Adaptive Hypertext and Hypermedia, User Modelling '94, Cape Cod.
12. Gruninger, M. & Lee, J. (2002) Ontology: Applications and Design. Communications of the ACM, 45(2), 39-41
13. Hubscher. R., Puntambekar, S. (2001). Navigation Support for Learners in Hypertext Systems: Is More Indeed Better? J. Moore et al. (Eds.) AI in Education, Amsterdam: IOS Press, Frontiers in AI and Applications, 68, 13-20.
14. Lambiotte, J.G., Dansereau, D.F., Cross, D.R. and Reynolds, S.B. (1989). Multirelational semantic maps. Educational Psychology Review 1(4), 331-367.
15. Mizoguchi, R., Bourdeau, J. (2000). Using Ontological Engineering to Overcome Common AI-ED Problems, International Journal of AI in Education, 11 (2), 107-121.
16. McGuinness, D.L. et al. (2000). An Environment for Merging and Testing Large Ontologies, KR00, Breckenridge, Colorado. April 12-15.
17. Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. International Journal of AI in Education, 10, 98-129.
18. Murray, T., Shen, T., Piemonte, J., Condit, C., Thibedeau, J. (2000). Adaptivity in the MetaLinks Hyper-Book Authoring Framework, Workshop Proceedings of Adaptive and Intelligent Web-Based Education Systems workshop at ITS 2000, Montreal, June 2000.
19. Okamoto, T., Cristea, A. (2001). A Distance Ecological Model for Individual and Collaborative-learning support, Educational Technology and Society, 4(2) April.
20. Okamoto, T., Kayama, M. and Cristea, A. (2001) Considerations for building a Common Platform of Collaborative Learning Environment, ICCEí01, Ed.: C.-H. Lee, 2, 800-807.
21. Sowa, J. F. (2001). Building, Sharing, and Merging Ontologies, http://www.jfsowa.com/ontology/ontoshar.htm