International Conference on Computers in Education

ICCE 2002

3-6 December 2002
Auckland, New Zealand

-------------------------------------------------------------------------------------------------------

Workshop on Concepts and Ontologies in Web-based Educational Systems

Program Committee

Lora Aroyo, The Netherlands (co-chair)
Darina Dicheva, USA (co-chair)

Jacqueline Bourdeau, Canada
Peter Brusilovsky, USA
Hugh Davis, United Kingdom
Vania Dimitrova, England
Judy Kay, Australia
Piet Kommers, The Netherlands
Kinshuk, New Zealand
Tanja Mitrovic, New Zealand
Riichiro Mizugushi, Japan
Toshio Okamoto, Japan
Sadhana Puntambekar, USA
Julita Vassileva, Canada
Pre-workshop proceedings

Preface 3

David Abraham and Kalina Yacef
XMLTutor - an Authoring Tool for Factual Domains, 5

Laurent Cimolino, Judy Kay
Verified Concept Mapping for Eliciting Conceptual Understanding, 9

Antonija Mitrovic and Vladan Devedzic
A Model of Multitutor Ontology-Based Learning Environments, 15

Judy Kay and Sam Holden
Automatic Extraction of Ontologies from Teaching Document Metadata, 23

Kazuhsa Seta and Motohide Umano
A Support System for Modeling Problem Solving Workflow, 27

Miguel A. Sicilia, Elena García, Paloma Diaz and Ignacio Aedo
LEARNING LINKS: Reusable Assets with Support for Vagueness and Ontology-based Typing, 35

Lora Aroyo and Darina Dicheva
Concepts and Ontologies in Web-based Educational Systems, 41

Riichiro Mizoguchi and Jacqueline Bourdeau
Theory-aware Authoring Environment - Ontological Engineering Approach, 49

Trent Apted, Judy Kay
Automatic Construction of Learning Ontologies, 55

For more information:

Lora Aroyo <l.m.arroyo@tue.nl>
Darina Dicheva <dichevd@wssu.edu>
Preface

Web-based Educational Systems (WBES) is one of the fastest growing and challenging areas in educational technology research. To meet the high expectations and requirements of educational community, a present challenging goal is the development of advanced intelligent WBES that adapt their behavior to the learner’s state of understanding. Concept-oriented (ontology-based) architectures come as a promising solution in the development of such systems. Conceptual (ontological) structures, such as concept maps, topic maps, and conceptual graphs have a great deal of potential for organizing, processing, and visualizing subject domain knowledge and for building learner models in WBES. Concept-based courseware employs conceptual domain presentation to link educational materials to a course structure. This allows for concept-based course sequencing that supports adaptive courseware generation, concept-based information retrieval, visualization, and navigation that help students to get oriented within a subject domain and build up their own understanding and conceptual association.

The goal of this workshop is to explore issues concerned with the design of concept/ontology-based WBES and to outline the state-of-the-art in the development of ontology-aware systems that facilitate the reuse, sharing, and exchange of knowledge and components with respect to both their creating and authoring. We hope that it will contribute to defining the path towards a new generation authoring strategies, which will meet the complex requirements of concept-based WBES authoring.

The benefits and implications of using ontologies in educational systems could be viewed from various angles. By providing common vocabulary for domain knowledge representation they can support interoperability of learning material, abstract representation of learners and user/group interactions, etc. Moreover, numerous general and specialized task and topic ontologies are already available and need to be shared, reused, and easily maintained. Finally, ontologies are expected to facilitate the process of WBES authoring. All these require development of new methods for knowledge organization and processing including combining, merging, reusing, and sharing of knowledge structures. The latter is closely linked to metadata (XML) and reusable components.

The papers accepted for presentation in this workshop are diverse and address many of the above mentioned issues thus giving a good view of what are the current tendencies in concept/ontology-aware WBES research.

An important research direction is related to the ease of construction and use of ontologies in WBES. The paper by Kay & Holden discusses an automatic extraction of ontological structures from teaching document metadata. It proposes a minimalist approach to metadata requiring the author to specify only the concepts, which each document teaches, requires, and uses. Apted & Kay describe a system, which automatically constructs an extensive ontology of Computer Science based on existing reliable resources. The system includes tools for querying the ontology (single- and multiple-concept queries and merging ontological structures) and visualizing the results.

Ontological knowledge representation facilitates building scrutable learner models. Abraham & Yacef’s XML Tutor uses XML as an ontological structure to represent an overlay student model, open for inspection by the student. This simple structure allows creating stereotypical models, which from one side are more representative as initial student models, and from another, can be inspected by the students for comparison with their own models. However, a critical problem in using concept maps for student modeling is related to verifying the maps before using them for reasoning about student knowledge. The paper by Cimolino & Kay proposes an approach to verifying concept maps for eliciting learner’s understanding of a domain. They have developed a tool that allows a student to construct a concept map, which is then checked against the teacher’s map and the student is prompted to reflect on and confirm or revise elements of his/her map.

With respect to the design and implementation aspects of WBES there is quite an extensive effort toward facilitating the reuse and sharing of educational components, knowledge and conceptual structures. Mitrovic & Devedzic present a model for building multitutor ontology-based environments in which Web-based intelligent tutors share ontologies, i.e. meta-knowledge, as opposed to sharing knowledge. Their goal is to create a unifying framework for enabling interpretability of multiple ITSs on the Web. In the same direction, Seta & Umano propose an ontology-based framework for planning of problem solving workflow in learning processes, where the domain independent problem solving tasks are modeled separately from the domain-dependent components (knowledge and resources) thus allowing their sharing and reuse for various domains. Sicilia et al.´s work focuses on the reusability of learning resources. They propose the concept of ‘learning link’ as an independent and reusable resource that can be annotated with terms defined in link ontology.
As a key direction to a new generation authoring strategy comes the ontology engineering approach for construction of theory-aware authoring environments proposed by Mizoguchi & Bourdeau. This is an effort towards building good learning support systems, well justified by basic instructional theories. In order to support system designers with theory-based help the authors have the ambitious plan to systemize learning / instructional / instructional design science knowledge from engineering point of views through ontological engineering.

In the same direction Aroyo & Dicheva propose an ontological approach towards a common authoring framework to formally describe the process of presentation generation in concept-based Web Information Systems (WIS). They suggest complete and clear separation of the content and the application related issues within WIS, which allows for better modeling of the application-related processes and for making accessible and retrievable external data in adaptive (user-oriented) WIS. The proposed approach provides a link between high-level author’s tasks and low-level developer’s functions and aims at supporting both WIS application authors and system developers.

Lora Aroyo
Darina Dicheva
XMLTutor – an Authoring Tool for Factual Domains

David Abraham and Kalina Yacef
School of Information Technologies
University of Sydney
kalina@it.usyd.edu.au

Abstract

The XMLTutor is an Authoring Tool that generically delivers personalised teaching on an arbitrary domain, which must be specified in Extensible Markup Language (XML). The tool was built to explore the consequences of using XML to represent the domain ontological structure in an Intelligent Tutoring System (ITS).

1. Domain Knowledge Module

Constructing domain knowledge can be a time intensive task [4]. Often it is difficult for a domain expert to translate their knowledge and thinking into a foreign format. A promising solution to this difficulty is the XML format. XML has a natural hierarchical structure similar to the contents section of a book. This allows XML to store both structural and semantic information about the domain. Mizoguhi [3] describes this set of XML tags more formally as a kind of level 1 ontology.

XML is used extensively outside the Intelligent Tutoring field as a data markup language and so many people would already be familiar with it. Furthermore, the language is very simple, making it an ideal format for non-computer experts to encode their factual knowledge in.

The XMLTutor can read in an arbitrary XML file and deliver a personalised tutorial on the material inside it. There are already XML descriptions of almost everything from Stock Prices to Geography (see Figure 1). With little or no work, a tutorial can be constructed on any of these areas.

![Example Domain Encoding](image)

Figure 1. Example Domain Encoding

The XMLTutor can do this because the knowledge is expressed in a common format. There is nothing immediately special about XML - if this knowledge were encoded in a semantic network then the equivalent Semantic Network Tutor would be able to automatically generate tutorials as well. Having said that, there are real advantages to using XML for knowledge reuse over other possible formats. This paper will attempt to highlight some of these advantages.

2. XML Properties

XML was created after years of experience dealing with the problems of other markup languages. It has a number of useful properties that overcome these problems [6].

XML is stored in plain text format. This makes it possible to view/edit XML in anything from a simple text editor to an integrated visual development environment. XML documents are also very readable and can typically be understood by inspection with a text editor. Binary formats are not as flexible and not as readable. Additionally, it is possible to represent binary data, such as audio and video, in XML using a base64 encoding.

Some markup languages have complicated grammars. For instance, in HTML, the <dt> tag can be closed by any one of </dt>, <dt>, <dd> or </dl>. XML has a very strict grammar – a <dt> tag must be closed by </dt> [6]. Furthermore, the tags must be completely and correctly nested. These restrictions make it simple to write a parser for XML. In fact, many programming languages already have libraries to parse XML – something that is not true of other domain encoding formats, such as semantic networks.

![Example of a small graph structure](image)

Figure 2. Example of a small graph structure

XML has great expressive power. Although Figure 1 labels the data in a strict hierarchy, XML also includes the ability to link to other tags or XML documents. For example in Figure 2 we can see that the node “London” has parent (London is a city but also a capital) and child nodes, and also is also linked to the node “The Thames” through the link “is crossed by”. Linking means that

---

1 Procedural knowledge is better represented by a more powerful format, such as an expert system or programming language.
XML can fully describe arbitrary graph structures (not only trees), where tags represent nodes, and links represent edges. This means XML is equivalent in expressive power to any database or semantic network.

3. Student Model

For each student, the XMLTutor maintains a separate instance of the domain knowledge, overlayed with some additional personalised information (see Figure 3). This information is stored in attributes of the XML nodes and includes the student’s number of correct and incorrect answers, as well as their confidence in their last answer.

![Figure 3. Example Student Model](image)

Although the Student Model used by the XMLTutor is simplistic, it contains the essential information needed by the Pedagogical Module to intelligently personalise teaching for each student. The model can easily be extended to include more information by annotating the XML nodes with additional attributes.

The XMLTutor allows the student to visualise and edit all aspects of their student model. This has two main advantages. Firstly, visualising the Student Model encourages students to reflect on their learning and to identify any areas of weakness. Secondly, the student may also correct any inaccurate information in the Student Model (see the Knowledge Relevance section for an example). This can help the Pedagogical Module deliver higher quality teaching.

The XMLTutor contains an XML viewer similar to the one built into the Internet Explorer browser. This viewer allows the student to explore different aspects of their model by expanding and collapsing whole trees of nodes. Alternative XML viewers, some of which are quite powerful, are already available in the Database community [1]. These viewers, like many other XML tools (parsers, editors, database converters etc), can often be plugged directly into an ITS. The existence of these free tools is one of the main benefits of using XML.

The XMLTutor allows the ITS designer to create stereotypical student models [5, 2]. The ITS designer can then supply a set of questions and answers that helps to classify new student in terms of the stereotypes. Therefore, when a new student uses the XMLTutor, their Student Model can be set to some combination of the existing stereotypes. This model should be more representative of the student than the alternative - a blank Student Model.

Students can inspect these stereotypes and compare them with their own model (for instance, how their knowledge of British Geography compares with a stereotypical European). These comparisons involve performing a sequence of basic Student Model operations such as addition, difference, multiplication and division which allow the student to make fairly complex comparisons with ease. For instance, the difference between two Student Models is calculated by taking the difference between attributes (e.g. confidence) in each node common to both models.

Because the stereotypes and student models are XML documents, these operations essentially involve a single traversal of the XML tree structure. This functionality is packaged as a small intuitive language inside the XMLTutor allowing the student to make arbitrarily complex comparisons with ease.

4. Pedagogical Module

The Pedagogical Module in the XMLTutor uses information in the Student Model, along with the domain knowledge, to deliver personalised instruction to a student. All sequencing of the instruction (topic selection) is done automatically using the structure of the underlaying XML. The module also generates natural language materials to present to the student.

Each XML node in the student model is placed in a priority queue. The Pedagogical Module repeatedly removes the node with the highest priority, delivers a question on it, and places the node back on the priority queue with an altered priority (based on the correctness and confidence of the student’s answer). Because nodes with higher priorities are more likely to be presented to the student, the problem of topic selection reduces to controlling the priorities of each node.

The basic parameters for computing the priority of a node are its correct, incorrect, confidence and relevance attribute values. By using these values, the Pedagogical Module is able to implement the four following topic selection principles.

Knowledge Relevance. Domain experts can associate a relevance value with each node in the initial domain knowledge. This value specifies the importance of the knowledge contained inside the node. The Pedagogical Module assigns a higher priority to nodes with a higher relevance value, meaning that important knowledge is more likely to be taught to the student.

Students can change the relevance value of any nodes in their Student Model. Therefore, after determining which parts of the domain are important for them, the students can influence the delivery of questions from the Pedagogical Module.
Exploration. After focusing on the important areas, the student needs to explore other parts of the domain. To this end, the priority of a node is partially based on the total number of questions the student has answered for the node (correctly or otherwise). When the total is small (i.e. the student has seen few questions on the node) the priority is assigned a relatively higher value, making it more likely the node will be selected as the next node to present to the student.

Reinforcement and Revision. The Pedagogical Module balances the exploration of new material with the reinforcement of material previously covered. With a complete student model, it is usually very easy to decide which pieces of knowledge need reinforcement and how urgently it is required. Nodes with higher incorrect values identify knowledge that the student has consistently failed to master. Nodes with lower confidence values highlight areas that the students know themselves to need revision in. Both of these attributes contribute to higher priority values.

Coherence and Learning by Association. As much as possible, the domain should be presented to the student in a series of coherent subtopics. This enables the student to learn not only individual concepts in the domain knowledge, but also associations between these concepts. The Pedagogical Module determines these subtopics through the structure of the XML encoded domain knowledge. The selection of a node from the priority queue increases the priority of all the nodes in its neighbourhood. A neighbouring node (one with some connection to the previous node) is then more likely to be selected next.

Natural Language Questions. Once a node is selected, the Pedagogical Module needs to present it to the student. This is done by automatically generating a question for the student to answer. The hierarchical and descriptive structure of XML makes it a straightforward task to describe a node in natural language.

The Pedagogical Module has a number of template natural language questions. These templates are instantiated by substituting actual tag names and data values from the XML structure. This template design is extendable, allowing the designer to easily add new question constructions.

5. Future Work

Generating questions is not the only use of an XML encoded domain. Students may also find it useful to treat the domain as a database by submitting it queries such as: What is the capital of England? The underlying XML structure makes it relatively easy to answer these queries. Indeed, XML has been used as a common representation format by the Database Community for some time now. The XMLTutor allows students to inspect the domain when they are visualising their student models. We are currently working on adding query functionality.

Converting XML to natural language is likely to be an important process in many fields. The XMLTutor contains a first attempt at implementing this conversion. Further work in this area will produce more interesting and readable conversions. This work is not likely to be carried out in the Intelligent Tutoring field, but, once done sufficiently well, it will become another free tool that comes with the choice of using XML.

6. Conclusion

The XMLTutor demonstrates the power of using XML to encode domain ontological knowledge. By defining a standard representation format, the program is able to deliver personalised instruction in any domain. Non-computing experts should find it much easier to translate their knowledge into the simple XML format. Once this ontology is built, the XMLTutor is able to automatically generate student models and instruct students in a way that maximises their personal learning.

References

Verified Concept Mapping for Eliciting Conceptual Understanding

Laurent Cimolino, Judy Kay
School of Information Technologies
University of Sydney, Australia, 2006.
{laurent,judy}@it.usyd.edu.au

Abstract

Concept mapping is a valuable technique for education evaluation. This paper describes a system which supports teachers in creating concept mapping tasks intended to capture the student’s understanding of the ontology of a small domain. A novel feature of our work is that the system verifies that the student intended the map elements that will be used to infer their understanding and misconceptions.

1. Introduction

Concepts maps [3, 7] have become a common tool for externalising learner conceptions of a domain. There are many available tools for learners to draw concept maps, such as those summarised at [2, 5]. Concept mapping has strong foundations in theories of learning and in empirical studies of brain activity [7]. The approach has many potential roles in education.

We are concerned with the use of concept maps as a mechanism for determining the way that a learner conceptualises a domain. From this, we aim to build accurate and detailed learner models of the learner's conceptual knowledge of a domain. Since our research focuses on scrutable learner modelling and strong learner control, the concept map is a natural tool for eliciting learner models.

In line with our philosophy of learner control, our current work explores approaches to building verified concept maps for the purpose of modelling the student's knowledge. We consider it critical to verify concept maps before using them as a basis for reasoning about the student's knowledge. This is because it is very easy for a student to accidentally link the wrong concept or omit a concept or a link from their concept map. Moreover, [7] emphasises that revision of concept maps is a normal and important part of the concept mapping process: he comments ‘it is always necessary to revise [the original map].’ Good maps usually undergo three to many revisions.’

This paper describes our system for verified concept mapping. Section 2 describes the interface from a student’s point of view, Section 3 from a teacher’s. Then we discuss the way that a concept mapping task can be designed to capture a student’s conceptualisation of the ontology for a domain. We also discuss the management of alternate ontologies and the role of this tool in the construction of detailed student models.

2. The student interface

Figure 1 shows an example of the interface as a student might find it at the beginning of a concept mapping task. In this case, the teacher has set up the task with a partial map to get the student started. We now give an overview of the interface.

The top tool bar shows the commands. The File menu has the usual commands to save, open, export and print concept maps. Edit allows the user to undo and redo actions. View manages the display of the graph: the user can view the graph with or without the grid, the rulers or the page layout. There is also a zoom. Shape supports organisation of the graph elements: aligning them in any direction as well as grouping and ungrouping. Extras sets the value of the grid and gives an overview of the graph in a pop-up frame.

The second tool bar supports short cuts. It also has the Analyse command, which the student selects on completing a task. This starts the process of checking the map after which the system asks questions which are intended to verify the map drawn. We will return to this aspect: it is core to the verification of the concept mapping process.

In the next row are icons for drawing the concept map. The button labelled with the arrow symbol is the default action. It allows the user to select, unselect and move elements of the map. The is used to add a new concept in the map and is used to add a new link.
between two concepts. Normally, the teacher will have created the concepts and links and these will be visible to the student in the left and right panels respectively. However, if the student really feels the need for additional ones, the interface does allow students to create new concepts and links so that they draw the concept map as they feel it should appear. As we discuss below, the system can only verify the use of concepts and links that the teacher had anticipated and made available to the student.

The very bottom bar of the interface is a status area. It provides an explanation of the mouse pointed element.

Figure 1. Example of start up map as presented to the student

Figure 2 shows an example of a concept map that might have been created by a student. We have used colour to indicate errors and will refer to these later.

Normally, the student would create the map in two stages. First, they would place the concepts on the map. This involves selecting them from the menu at the left and placing them. Then, the student would typically go through a stage of moving the concepts around on the map, getting the layout to really reflect the structure desired. The concept mapping process requires that the student should place higher level concepts higher on the map and more specialised ones lower. Similar level concepts should be at the same level. Also, concepts that are thought of as similar should be placed together. It is also desirable to ensure that the layout is pleasing. As this description suggests, this layout process is quite cognitively demanding and will typically require many revisions and refinements. Essentially, this stage of the process involves externalising part of the student’s ontological understanding of the domain as it is supposed to capture the hierarchical relationships between concepts, similarity relationships and symmetry relationships. It also involves laying out the concepts so that it will be easier to connect the concepts that will be linked. So, concepts that are related will need to be placed in reasonable proximity to each other.

In the second stage, the student completes the rest of the ontology externalisation. This is the construction of the propositions. For example, the initial map in Figure 1 has the proposition Plants have roots. In general, propositions constructed with this interface have two concepts connected by a link. During this stage, it will be common for the student to move the concepts around to reduce crossing of link lines and to improve the map appearance.

Figure 2. Example of a map that has been constructed by a student

When the student thinks their map is complete, they click the Analyse button. First the program checks if each concept has been included in the map. If not, the analysis stops and the program shows a list of missing concepts. Once all the required concepts have been used, the Analyse phase checks the other aspects of the map. First, and most important are the propositions that are the essence of the map. For example, the top part of the map in Figure 2 represents the proposition Plants have Roots.

If the student’s map is missing a proposition that the teacher expected in the map, this analysis phase will result in a message which hints about the missing proposition. This type of message is illustrated at the top of Figure 3. The actual details of these messages are entirely
controlled by the teacher who sets up the concept mapping task. It is quite feasible to allow for different understandings and associated propositions. It is also the teacher’s decision to have more or less checking of the map and corresponding numbers of messages.

3. The teacher interface

The teacher uses an interface like that shown in Figure 4. It is very similar to the interface that the student will see. However, there is an additional Teacher menu. This supports special functions needed to create a new concept mapping task. These include definition of the parts of the teacher’s map to be made available to the student at the beginning of their mapping activity. It also provides access to the parts of the interface needed for defining the analysis process for a map.

We now describe the process a teacher would go through in order to create a map shown in Figure 4.

Figure 3 also provides examples of system messages which indicate the inferences the system draws from the concept map. This is the set of statements listed under the heading Sentences to save. The first block of these is the propositions that the system has recognized and which the teacher has specified for saving. For example, the first indicates that the result of the mapping activity will include a record that the student knows the concept Plants have Stems. The fourth last statement, Student knows the level rule Flower components, is an example of an inference about the way that the student has laid out the concepts at the same level to indicate that they are similar. The last three statements all indicate that the layout of the map has been used to infer that the student understands the hierarchical relationship between the sets of concepts.

The reason we have included this screen is so that the student can see the complete set of inferences that have been made about their map. At this stage of the system development, we expect to need to refine the details of the display in order to ensure its effectiveness.

3.1 Creating concepts and links

The first stage in creating a concept-mapping task is to define the concepts that will appear at the left of the screen and the links that will appear at the right. Selecting the New button above either list brings up a screen for creating a new concept. Creation of links requires the teacher to define whether the link is uni-directional as in the case of links like has or bidirectional as in the case of links like synonymous. The interface also requires the teacher to provide additional information for the longer term use of the student modeling: the concept or link must have a creator and an explanation. The latter is potentially useful for capturing some of the subtlety of the teacher’s reasoning in the choice of a particular term. For example, it may be that one term was used in the textbook or
another was a common source of difficulty in the past. Perhaps there is an unusual term that the teacher prefers.

Essentially, this is the stage at which the teacher is defining the vocabulary underlying their ontological understanding of the area covered by the concept mapping task. Any reasoning that was important in defining the vocabulary can be captured at this stage. The Edit menu allows the teacher to later edit these concept and link definitions as needed at a later stage.

3.2 Building up the mapping task

The teacher can perform the bulk of this process in the same way as a student drawing a concept-map. We wanted this to be the case so that the teacher would be able to have the same look and feel as the student would have when doing the task.

The teacher can also create propositions using the New Proposition button in the tools bar. This is essential for the definition of propositions that constitute alternate conceptions or ontological views of the subject. The teacher needs to define these so that the analysis phase can deal with incorrect or other alternate propositions that the teacher expects students to create. This aspect is important in enabling the verified concept mapping system to elicit student models for such misconceptions or alternate views. Figure 5 shows the screen that can be used to define new propositions. This provides the complete list of current concepts at the leftmost and rightmost panels. In the middle is the complete list of links. Selecting one item from each of these enables the teacher to define any proposition that is possible for the current set of concepts and links.

3.3 Creating analysis actions

Each link on the visible map can be used to select a proposition. Typically, the teacher would go through each such proposition in their own map and the additional propositions they have defined and, for each, define the actions that should occur in the analysis phase. The Teacher menu gives access to a complete list of the propositions. This provides access to the interface screen for defining these actions as illustrated in Figure 5.

Each new proposition has two default processing actions. These are illustrated at the bottom of Figure 5, labelled with the number 1. The first indicates that if a student correctly creates this proposition, the output of the concept-mapping task will result in saving the proposition as known. The second is to automatically ask the student to consider amending their map if the proposition is missing from their map. The default text for this is shown in the figure. As the interface has a text box for these messages, the teacher can choose to alter them to any arbitrary text of their own choosing.

Figure 5. Creating arbitrary propositions and adding analysis actions to them

An example of the screen for defining a misconception is illustrated in Figure 6.

The interface supports creation of analysis rules for the vertical layout of the concept map. These are created in much the same way as the processing rules for propositions. There are two types of rules, one for checking that a set of concepts have been placed level with each other and the second checking for a hierarchical placement with one set of concepts above another set.

Figure 6. Creating a proposition to reflect a misconception

A level rule has only a name, a set of concepts and its processing actions. The teacher needs to select the level rule button in the tool bar. Figure 7 shows a set of level rules. For example, it indicates there is a rule called Plant components, marked 1 in the figure, and it has the three concepts, Roots, Stems and Leaves, marked 2 in the figure.
To create a new level rule, the teacher selects the New in Figure 7 and is presented with the window in Figure 8. This operates similarly to the analysis rules for propositions. The first default rule saves an indication that the student correctly built a map with this level rule satisfied when this is the case. Where just one or two concepts are missing, the default rule asks the user to check the map, with the teaching providing the wording of the question. The teacher can alter the message and processing action. The buttons at the bottom of the window also allow the teacher to created additional rules. For example, there might be a Save action as well as an Ask.

The third, and last, rule type checks for hierarchical structures in the map. A hierarchy rule has a name and two set of concepts, those concept(s) above and the ones below. Figure 9 shows the interface for creating such rules. Initially, the set of concepts all appear in the panel at the right. The teacher selects those for the appropriate windows at the left, so specifying the set of concepts which should appear above those specified as belonging below. The processing operates similarly to the cases already described.

In the processing, the program determines the relative positions as follows. It finds the smallest rectangle containing all the selected concepts. Then it attempts to separate that rectangle into two by cutting horizontally so that every concept in the top rectangle is in the above-set and all concepts in the bottom rectangle belong to the below-set. If it succeeds in finding such a split, it judges the student’s map to pass on this rule. This is a quite generous interpretation which allows the student considerable flexibility in the actual layout.

4. Implementation

The system has been implemented in Java and requires at least Java 1.4. It is based upon the JGraphpad\(^1\) open source project and uses JGraph\(^2\) Swing.

---

1. http://www.jgraphpad.com
5. Discussion

Our system allows the student to construct a map and then, once they decide that it is complete, they can indicate this at the interface. The system then checks the map and prompts the student to check selected elements. This means that it helps the student verify that they really are content with the final form of the map. This phase of the mapping activity can help the learner check that they have not made careless mistakes. It also helps them think about additional links and conceptual relationships that they actually do know about but may have failed to include in the initial map. Since concept mapping is such a cognitively demanding activity, it is highly likely that the learner will forget some of the map elements that they actually do know about. This prompting phase helps them to return to the task and focus on elements they have forgotten.

Another important part of our prompting process is that the system can help the learner examine parts of the map that suggest misconceptions. Where the map has been accidentally created to suggest a misconception, the prompting phase can help the learner correct that. Equally, if the learner really does hold the misconception, the prompting phase ensures that the final map has captured this misconception. This is somewhat similar to the approach of comparing student and expert maps as in [1].

We might consider the student's completed concept map to be a form of student model. However, we prefer to use it as one source of information about the student's conceptual knowledge and this can be kept in our generic user model framework [4]. This can then be combined with various other sources of information about the learner, such as their performance on various other learning activities.

It is in this spirit that we include a summary of all the inferences that the system derives from the mapping activity. This means that the student can scrutinise this list to see just what the system will regard them as knowing as well as aspects that will be modelled as beliefs the student holds although the teacher regards them as misconceptions. Such an interface invites the student to reflect on such misconception.

Our interface has tried to remain true to the spirit of the original work on concept mapping by [6]. The student is able to create their own concepts and links if they wish. This means that they are not constrained. At the same time, if the student does keep to the set of concepts provided by the teacher, they can gain feedback on how that teacher had expected them to draw the map and some of the essential elements expected.

Essentially, the interface supports the student in externalising their understanding of the ontology of the domain of the concept mapping task. The teacher’s set of defined concepts and links as well as the initial parts of the map should help ensure that the student takes a perspective in line with that the teacher intended. Once the student has completed their own map, the analysis phase enables them to see the differences between their ontology and the one the teacher expected. The questions that the student is asked to reflect upon their own ontology and the elements that differ from the teacher’s.

6. Conclusions

We have constructed a system for eliciting student understanding of a domain. We believe that our approach has considerable promise for modelling student’s personal ontologies. User trials will be needed to assess the effectiveness of the interface and the approach.

References

A Model of Multitutor Ontology-Based Learning Environments

Antonija Mitrovic\textsuperscript{1} and Vladan Devedzic\textsuperscript{2}
\textsuperscript{1}Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
tanja@csc.canterbury.ac.nz
\textsuperscript{2}Department of Information Systems, FON - School of Business Administration
University of Belgrade, POB 52, Jove Ilica 154, Belgrade, Yugoslavia
devedzic@galeb.etf.bg.ac.yu

Abstract

The paper proposes the M-OBLIGE model for building multitutor ontology-based learning environments. We show how the model can be applied to tutors in the database domain. The proposed model can be used as a framework for integrating multiple tutors on the Web. We also illustrate the process of developing the ontologies for an existing system.

1. Introduction

Once a team of ITS developers builds a successful ITS that helps students learn a part of some knowledge domain efficiently, chances are that they may want to extend and upgrade the system later on. If the knowledge domain is complex enough, chances are also that the developers may want to build other tutors to cover other parts of the knowledge domain. Several examples in the history of ITS design and development illustrate this point, e.g. [1, 2, 12, 14, 22].

In such cases, developers will want their tutors to reuse parts of knowledge built in other tutors they have developed. Moreover, two different teams may develop two different tutors that partially overlap in expertise. Therefore, the need for automatic knowledge sharing, reuse, and exchange among several different tutors in the same complex domain is highly likely to arise. This is especially the case in Web-based education, where different tutors may be distributed across a number of sites.

This paper analyzes the role and cost-effectiveness of using ontologies in such settings. We first briefly look at some relevant ontology-related work in the domain ITS in Section 2. We present the model of a Multitutor Ontology-Based LearnInG Environment (M-OBLIGE) in Section 3. In Section 4 we discuss how that model would be applied in the context of database tutors [14] and how the architectures of these systems would change in order to accommodate the use of ontologies. We discuss the initial steps we took in making ontology-related enhancements to the database tutors in Sections 5, 6, and 7. The last section contains conclusions.

2. ITS and ontology-related work

The way we consider ontologies in our work is best described by Swartout and Tate's definition: ontologies provide the necessary armature around which knowledge bases should be built [21]. In other words, ontologies compose the necessary reference knowledge for making two or more related intelligent systems knowledge-consistent. Our emphasis is on the infrastructure role of ontologies for knowledge sharing among related Web-based educational systems. We put our research in the context of the emerging Semantic Web, a huge network of machine-understandable and machine-processable human knowledge, not just ordinary information [4, 8, 9, 10, 13]. The Semantic Web is expected to provide explicit representation of the semantics of data in the form of various domain theories stored on many Web-servers as a myriad of shareable ontologies, as well as advanced, automated, ontology-supported, and agent-ready reasoning services (http://www.semanticweb.org/).

The most notable work in the ITS community related to the development of educational ontologies, coming from the Mizoguchi Lab at Osaka University, Japan [17, 18], and from Tom Murray [19]. Mizoguchi et al. point out that the ITS ontology as a whole consists of domain ontology, which characterizes the domain knowledge, and task ontology, which characterizes the computational architecture of knowledge-based systems [17]. They have also made an important contribution to the hierarchy of ontologies in the domain of education, and have studied how the use of ontologies can contribute to the architecture of ITS and ITS shells and authoring tools. Mizoguchi and Bourdeau have noticed many limitations of current authoring tools in general, all of which pertain to tools for building Web-based educational systems as well [18]. Murray has defined the important topic ontology, based on topic types (e.g., concept, fact, principle,...), topic link types (e.g., is-a, part-of,
prerequisite, context-for,...), and topic properties (e.g.,
importance, difficulty,...) [19]. More recently, Devedzic
has studied educational services and servers in the context
of ontologies on the Semantic Web [6].

3. The M-OBLIGE Model

Two or more Web-based ITSs can refer to a common,
shared part of their knowledge as in Figure 1. Note that
both tutors (App1 and App2) have their own private
knowledge and reasoning mechanisms. The Web pages
corresponding to any single tutor in this scenario must
contain pointers to ontologies the tutor uses as its meta-
knowledge. Such semantic markup of the tutor's Web
pages enables an external agent or application to
recognize the "armature" part of the tutor's knowledge.
The ontologies themselves are stored in a machine-
readable form at possibly different locations on the Web.
They can point to each other as well, since the knowledge
from one ontology can refer to the knowledge in another
ontology. For example, the ontology of, say, fraction, can
refer to the ontology of number, which in turn may be
stored at another location on the Web.

Every such ontology should provide a set of
knowledge terms, including the vocabulary, the semantic
interconnections, and some simple rules of inference and
logic for some particular topic or service [Hendler, 2001].
Interoperability and knowledge sharing between different
educational applications can be achieved by using
appropriate languages for representing ontologies and
educational contents and services. Current trends in Web
technology suggest that appropriate representation
languages include XML, XML Schema, RDF, and RDF
Schema languages [4], all developed under the auspices of
WWW Consortium (http://www.w3.org/XML,
http://www.w3.org/RDF). For developing ontologies,
higher-level languages and development environments
built on top of these four are a good choice. Some of the
most popular examples are OIL and DAML+OIL [11],
and Protégé-2000 [9]. They let the developer develop an
ontology using highly interactive, graphical
environments, and then convert it automatically to a
language from XML/RDF families that ensures semantic
interoperability and knowledge sharing on the Web. M-
OBLIGE model follows that design philosophy.

We want to stress that this model of knowledge
sharing significantly differs from Web-based
collaborative learning environments, shared knowledge
bases, and shared student models. What is shared here is
meta-knowledge, i.e. the static part of the tutors' knowledge
that makes each tutor aware of the knowledge
teaching the implemented ER model. On the other hand, the
ontologies of data models and databases are general
enough to be used by other DB tutors and other intelligent
DB applications alike. Ontology processor will help the
pedagogical module to make instructional decisions and
enable external agents find out about KERMIT’s
instructional focus.
4.1 Ontological Engineering of Database Tutors

In the course of developing the initial ontological support for database tutors, we developed the ER modelling ontology for KERMIT, as well as the data model ontology that KERMIT can share with other database tutors. To develop these ontologies, we used Protégé-2000 ontology development tool [9]. Protégé-2000 can store ontologies in different formats, one of them being RDF Schema – an important language for building Semantic Web applications, providing semantic interoperability between applications, and storing ontologies in a machine-understandable form [10].

In developing the ontologies, we used the following guidelines suggested by the Semantic Web community and discussed in detail in [6]:

- Planning for expected uses of the ontology. In our case, the expected applications to use the ontologies include (but are not limited to) the suite of database tutors mentioned in Section 4.
- Start by building small, largely incomplete ontologies, and let them grow incrementally and iteratively over time, as opposed to working on an elaborated, complex conceptual design of ontologies for a long period of time before actually deploying them. This way we wanted to avoid the "analysis paralysis", i.e. the danger of just thinking about something forever, without putting it to life [5].
- Test the initial ontology by using it to build a knowledge base. We used the knowledge already built in KERMIT, NORMIT, and SQL-Tutor, this time in order to instantiate our ontologies.
- Iterate over the above steps in order to revise and upgrade the initial ontology incrementally.

4.2 KERMIT's Local Ontology

KERMIT teaches the Entity-Relationship (ER) model, hence its local ontology includes ER modelling ontology. That ontology is useful for many other applications, including the suite of database tutors. We envision extensive reuse of the knowledge built in the ER modelling ontology, as well as its several versions and implementations. Figure 3 illustrates a part of the hierarchy of the concepts in the ER model that we have derived starting from database design domain theory (e.g., see [7]).

Figure 2. Ontology extension for KERMIT

4.2.1 ER Construct

KERMIT teaches the Entity-Relationship (ER) model, hence its local ontology includes ER modelling ontology. That ontology is useful for many other applications, including the suite of database tutors. We envision extensive reuse of the knowledge built in the ER modelling ontology, as well as its several versions and implementations. Figure 3 illustrates a part of the hierarchy of the concepts in the ER model that we have derived starting from database design domain theory (e.g., see [7]).

Figure 3. Partial hierarchy of concepts in the ER model

Starting from such concept hierarchies, we developed the ER modelling ontology to the level of detail that allowed for suitable redesign of KERMIT's knowledge base to include ontological support. Figure 4 shows a screenshot of our Protégé-2000 implementation of the ER modelling ontology. Not all parts of the ontology can be seen in the screenshot, but two of the most important domain concepts – ER construct and ER attribute – are shown explicitly.

After developing the ontology, we exported it into a set of HTML pages for easier reference. Figure 5 shows an example of such a HTML page, representing the concept of weak entity from our ER modelling ontology. (the same concept is also visible in Figure 4). Each concept in an ontology developed by Protégé-2000 can be represented by such an equivalent HTML page, although in the case of a more complex concept, the page itself can be rather long.
5. A Shared Ontology for a Database Tutors Suite

Note that in the domain of database design ER model itself is only a special case of the more general concept of conceptual data model, and there is a still more general concept, the data model concept. We recognized these relations in developing our ontologies, and realized that other database tutors (as well as other applications) will need to rely on such more general ontologies in order to reuse and share common knowledge more efficiently.

Some of the most important concepts and relationships of the data model ontology, as well as widely accepted terms for representing them, are depicted in Figure 6. When we developed that ontology in Protégé-2000, we could easily converted into a set of HTML pages as we did for ER modelling ontology. Figure 7 shows the topmost HTML page in that representation. Figure 8 shows more details.

---

**Figure 5.** HTML representation of the *Weak Entity* concept from ER modelling ontology

**Figure 4.** Initial implementation of ER modeling ontology in Protégé-2000 includes *ER Construct* and *ER Attribute* hierarchies (left pane)
Since M-OBLIGE prefers XML/RDF representation of ontologies in order to make them Web-ready and agent-ready, we converted our ontologies into RDF format. Figure 9 shows a part of data model ontology in that format. Revisiting Figure 2, it is important to note that data model ontology in RDF format can be stored anywhere on the Web. It suffices for each database tutor in the suite to know the location of data model ontology and to include an RDF parser in its ontology processor in order to reuse the knowledge of data models encoded in that ontology. Moreover, marking up the Web page corresponding to a database tutor with a pointer to data model ontology and storing the markup with the Web page enables external applications and intelligent agents.
Conclusions

There is a growing need for applying ontologies in the field of ITS. Artificial tutors often need to exchange and reuse their knowledge, and ontologies provide infrastructure for that. The M-OBLIGE model proposed in this paper is our first step towards a unifying framework for enabling interoperability of multiple tutors on the Web.

References

Figure 9. A part of RDF representation of data model ontology

Automatic Extraction of Ontologies from Teaching Document Metadata

Judy Kay and Sam Holden
School of Information Technologies
University of Sydney, Australia
judy@it.usyd.edu.au

Abstract

SITS (Scrubtable Intelligent Teaching System) is designed to make use of existing learning items in flexible and effective learning interactions. The reuse of pre-existing resources is important since creating new learning resources is a time consuming task which requires a skilled author. The Internet also provides a large number of resources for reuse. A major hurdle for reuse is metadata, especially epistemological metadata since different teachers or courses may have different ontologies for a given domain. SITS takes a minimalist approach to metadata. It requires only that the teacher constructing the teaching environment should define document metadata specifying the concepts which each document teaches, requires, and uses.

1. Introduction

Reusing available teaching resources is valuable on several counts. First, the resources are time consuming to create. Perhaps more important is the fact that of the best resources are produced by extremely knowledgeable and talented authors. The World Wide Web is potentially a source for a large number of teaching resources for a diverse range of topics. Reuse of those resources should enable faster creation of a course, with the potential for higher quality and a richer coverage of areas.

For example, when creating a course to teach the C++ Standard Template Library, there are many high-quality and varied teaching resources which can be reused. These range from the “Standard Template Library Programmer's Guide”[1], to introductory tutorials such as “An introduction of STL for beginners”[2], and entire books such as “Thinking In C++”[3].

The potential value of these existing resources is increased if the teaching system has a domain ontology. This can serve as the foundation for the mark-up of those documents with metadata describing their contents. This can then be exploited by a teaching system which has a model of the student’s knowledge and their learning goals.

For example, in the C++ Standard Template Library domain a student may wish to learn about the unique algorithm. Suppose the teaching system has an ontology capturing the relationship of iterator and remove as prerequisite knowledge for unique. Consider the case where the system believes the student understands iterator but not remove. It can create a teaching plan starting with remove and then move on to unique.

The creation of such an ontology is non-trivial as it involves enumerating the concepts and relationships between the concepts being taught. Many methodologies have been used for representing such domain knowledge, ranging from rules [4] to genetic graphs[5]. In general, a domain expert must create the ontology in terms of the representation approach.

Along with this ontological metadata, SITS also uses more general metadata, such as the document author. This metadata is dealt with differently since it can reasonably assumed that it would be accessible from a conventional metadata description of a document. SITS teaching strategies use information about the author and the style of the document. Metadata about other properties of documents could equally well be used, as SITS treats them as labels without attaching any semantic knowledge.

2. Variable Teaching Strategies

Teaching strategies in SITS are separate modules of code. The role of the SITS teaching strategy is to select the documents that would best suit the student to meet their current learning goal. The teaching strategy controls the interface with the learner, generating all material presented to them.

A teaching strategy might aim to introduce new ideas in terms of the minimum number of additional concepts. This means that the material presented should be highly focussed on a single concept that is needed to teach the current learning goal. Only when this has been learned, would it introduce the next new concept on the path to the learning goal. By contrast, another teaching strategy would favour introduction of several new concepts at once. There are other dimensions of the teaching strategy. For example, one teaching strategy might favour documents which teach in terms of examples if such are available. On another dimension, another might operate in the context of a visualisation tool to allow the student to navigate the document space manually.
3. SITS Approach to Creating Ontologies

In SITS, the underlying ontology is constructed from the document metadata. A course in SITS requires the definition of the vocabulary of concepts. Each document in SITS has metadata specifying the concepts that the learner needs to know if they are to understand the document. These are the **prereqs**. It also requires definition of the concepts the document teaches. These are called **shows** metadata elements. Finally, SITS allows the teacher to provide metadata for the concepts that the document **uses**. These three types of metadata, **prereq**, **shows**, and **uses** are the only annotates that a teacher needs to define for a document which is to be made available within a course.

If we take a set of documents from a single source there is an inherent ontology. For example, in a book, one chapter will typically use concepts introduced in earlier chapters but will not use concepts introduced in later chapters of that book. The document metadata is used by SITS to extract this ontological information. Note that some of this ontology is due to the nature of the domain being taught. Other parts are due to the author’s idiosyncratic and personal ontology. It may be that two expert authors will disagree on the nature of the ontology in some aspects of their area of expertise. Equally, different authors will have different understandings of what is most important and just how things are related within the domain. All these factors will contribute to the existence of different ontologies underlying the sets of documents created by different authors.

When several sets of documents are put together a merged ontology is created based on the concepts the teacher has defined.

The ontology is dependent on the concept vocabulary defined by the teacher. The metadata created for each document defines the relationships between the concepts in the vocabulary. The relationships are inherent in the documents but are interpreted by the metadata creator and restricted by the concept vocabulary used. The teaching strategy being used will also affect the ontology used.

We illustrate this with a simple example where we consider two documents with the metadata shown in Figure 1. Suppose we have one teaching strategy which treats uses-metadata in the same way as prerequisite-metadata. This would result in a different ontology from that generated by teaching strategy which did not treat uses-metadata in such a restrictive way.

Now suppose that a student wants to learn about the **unique** algorithm. SITS examines the metadata of our two documents which show **unique**.

Further suppose that SITS believes the student understands the concepts shown in bold in Figure 1.

![](image1.png)

**Figure 1. Metadata of Two Documents**

The teaching strategy which treated uses concepts as non-essential could select either document to use to teach the **unique** concept. And the teaching strategy which treated uses concepts as if they were prereqs would need to first teach the student those concepts; in effect, the choice of documents would be a choice between two ontologies shown in Figure 2.

![](image2.png)

**Figure 2. Two Resulting Ontologies**

SITS represents the student knowledge using an evidence based student model[6]. The concepts used in the document metadata are the items for which evidence of knowledge is collected. SITS uses a very simple approach to building the user model: after they have viewed a document, SITS asks the student to self-assess how well they believe they understand each concept related to that document. This information is stored in an evidence list and resolved to a number on a scale of 0-5 (not known – known) when the information is needed.

4. Separate Metadata

Keeping the metadata separate from the documents, SITS allows the same documents to be used in multiple courses. The metadata of one document in different courses can be different. This gives the course constructor the freedom to interpret documents as they see fit. It is quite possible that they may even attach metadata which captures a quite
different ontology from that intended or anticipated by the original author.

Figure 3 illustrates this. Two teachers have used SITS to create their own courses, one shown as shaded elements and the other as white. Both courses use some of the same documents. These external documents are represented by the collection of pages in the centre of the figure. Those documents with thickened outline are used by one course, and the documents with shaded backgrounds by the other. Some documents have both a thickened outline and a shaded background; they are used in both courses.

**Figure 3.** Two courses sharing a pool of external documents resulting in differing ontologies

The course concepts are represented by the large lists next to the document collection. They are also marked with a shaded background or thickened outlines to indicate which course is which.

The internal metadata for each course is represented by a list linking the documents to the concepts. Again shading and outline thickness is used to indicate which course the metadata belongs to.

The two courses shown have some overlap in the external documents they use. The metadata for each course links the documents and concepts together and forms a different ontology for each course. The ontologies are different because they are based on different sets of documents and they have different metadata.

The final ontology is influenced by three factors:

The author of the original documents will have built in assumptions and expectations about what is important and what is not. The author will have built into their documents the domain ontology as they saw it, and this ontology will be reflected in the final collection of their documents.

The teacher influences the final ontology as well. The choice of concepts to model, and the metadata created for each document provide the teacher with considerable influence on the final ontology.

Finally, the teaching strategy influences the ontology in the way it interprets the metadata, as explained in Section 3.

5. Other Systems

Many successful systems have used a concept-document model in which documents have prerequisite and learning outcome metadata. Three well known, representative examples of this are ELM-ART[7], Interbook[8], and AHA[9]. SITS differs from these in that it aims to reuse arbitrary existing material, and extract the ontology from minimalist metadata.

ELM-ART is a WWW based ITS for the Lisp programming language. ELM-ART has a textbook and reference manual containing links to concepts. These are more detailed than the `prereq`, `uses`, and `shows` in SITS. For example, a concept can be ‘introduced’, ‘presented’, or ‘summarized’. These concept links are not used to derive the course ontology. Instead, it uses a predefined conceptual network containing all the concepts and their inter-relations.

Interbook is a Web based electronic textbook broken up into structured units. Each unit is tagged with ‘prerequisite’ and ‘outcome’ concepts, which are equivalent to `prereq` and `shows` in SITS. Interbook uses the tags to provide links to a glossary, and also links from the glossary to relevant units. Links to prerequisites are also provided if the student requests them. A ‘Teach me’ button takes the student to the unit that Interbook considers the appropriate next step for that student. Interbook differs from SITS in that its textbook approach, with next, previous, and up links. More importantly, it makes use of specially authored content.

AHA is a generic adaptive hypermedia system which uses a pre-processor to filter content fragments in HTML documents. Each HTML document contains a list of prerequisite concepts and ‘known’ concepts, which are equivalent to SITS `prereq` and `shows` metadata. Like Interbook, AHA requires specially authored content.

SITS is most similar to Interbook. The main differences are that SITS is designed to reuse existing content from multiple sources. Once the concepts are defined, adding a document requires only inputting the URL and checking the appropriate metadata checkboxes. SITS uses that metadata to explicitly create an ontology for the learner.

6. Conclusions

SITS metadata is the representative of the documents in the system. In combination with the teaching strategy, it is used to infer the ontology for the domain.
In addition, SITS makes use of other metadata, such as the author of a resource and the style of the resource (for example, reference, tutorial, or example). In the present implementation, the author and style are used. They are treated as text labels. Although SITS does not actually know what a ‘reference’ style is, it can exploit this information. The learner model represents whether the learner prefers documents whose style is ‘reference’. SITS consults the learner model and matched the learner preference with the document metadata.

Such metadata is handled by conventional mechanisms from an metadata from XML representation of IMS Learning Resource Metadata[10]. A simplistic metadata server is included in SITS to supply this metadata for resources which do not have an existing source. This is not required. If a document has no external metadata, SITS will still function. Teaching strategies which weight documents based on external metadata will simply use a default weighting for such documents.

By contrast, the metadata which relates the resource to the concepts being taught is part of SITS. Its creation is the essential task for adding a new resource to a SITS teaching environment. It provides the mechanism that relates resources to concepts. In general, a SITS learning environment is created by defining the set of concepts before any documents are added to the environment. A major attraction of this approach is that the addition of the hundredth or thousandth resource is as easy as adding the first.

The one change that one expects over time is that teachers will become aware of the need for additional concepts after they have marked up some of the documents. SITS allows the creation of new concepts at any stage. This poses problems since such concepts could not be part of the metadata of documents already in the learning environment. Ideally, the teacher would revisit the coding of all existing documents to ensure that new concepts have been added to their metadata where this is relevant. However, at worst, the new concepts can simply be used for future document’s metadata. Concept and metadata information can be time stamped at creation and modification time. Those time stamps can be used to provide the user with a summary of the items that may need modification when a new concept is added.

Formative evaluation of SITS is being performed with the quite substantial STL domain. It remains to perform experiments to evaluate the quality and effectiveness of the ontologies generated by SITS. The SITS approach seems promising for the automatic and pragmatic construction of ontologies. It makes use of modest collections of metadata that a teacher creates for each of the documents that are to be used or reused for a rich teaching environment.

The ontology creation in SITS occurs on multiple levels. The original authors, the teacher, and the teaching strategies are all involved in its formation. SITS takes this information and in association with each learner’s knowledge and preferences automatically constructs an ontology for each learner.

References

A Support System for Planning Problem Solving Workflow

Kazuhsa Seta and Motohide Umano

Department of Mathematics and Information Sciences, Osaka Prefecture University, 1-1, Gakuen-cho, Sakai, Osaka, 599-8531 Japan
{seta, umano}@mi.cias.osakafu-u.ac.jp

Abstract

Novice learners in a problem solving domain build up their own understanding of its target area, make plans to solve a problem based on their understanding and perform the plan. Our goal is to develop a system for facilitating those processes and the reuse, sharing and inheritance of expertise acquired with great efforts.

In this paper, we mainly discuss the issues to realize the former goal. Firstly, we analyze novice learners’ work to clarify what support is required for them to perform their work. Secondly, we propose a framework sufficient for performing their work. Thirdly, we discuss a process ontology to encourage the interaction between meta cognition activities and cognition activities. And finally, we illustrate the system which supports learners’ work in problem solving.

1. Introduction

Novice learners in a problem solving domain build up their own understanding of its target area and make plans to solve a problem based on their understanding. Our goal is to develop a system for facilitating those processes and the reuse, sharing and inheritance of expertise acquired with great efforts.

The basic philosophy underlying our approach is to encourage learners’ awareness and reflection of their work so that they develop strong problem solving skills. Ontology is a key technique to achieve this goal [1].

In this paper, as a first step, we mainly focus on the issue of supporting learners’ planning processes and describe our framework for effective and efficient planning of problem solving workflow.

2. Task analysis of learners’ work

In this section, we analyze learners’ work to provide useful support to learners.

2.1 Learners’ work in problem solving

Consider an example of a learner who is not very familiar with Java and XML programming and tries to develop an XML based document retrieval system. He/she explores web space, gathers useful information and plans learning and problem solving processes. In our scenario, novice learners in a problem solving domain try to gather information from web sources, investigate and build up their own understanding of the target area, make plans to solve the problem in hand and perform the problem solving and learning processes. Needless to say, a complete plan cannot be made at once but should be detailed gradually by iterating spirally those processes applying the ‘trial and error’ approach as shown in Fig. 1.

2.2 Task analysis by knowledge decomposition

Problem solving knowledge, in general, can be decomposed into a task-dependent but domain-independent portion and a task-independent but domain-dependent portion. The former is called task knowledge, e.g. knowledge on diagnosis, planning, learning and so on, and the latter domain knowledge, e.g. knowledge on device models, mathematics and so on [2]. This idea is well known in knowledge engineering field as “knowledge decomposition.”

By analyzing learners’ problem-solving work using this idea, we understand that this work is very complicated, that is, a problem solving task of planning is decomposed into a type of planning task and target domain. The domain, in our case, consists of learning and problem solving activities. Furthermore, learning and target problem solving activities are decomposed into a type of learning and target problem solving and their target domain. Since this nested structure...
of learners’ tasks puts heavy loads on the learners, they tend to fall into utter confusion and lose their way.

Based on the above task analysis, we propose a reference model of learners’ work as a detailed version of the rough model shown in Fig.1. (see Fig. 2).

Problem solving processes get a version of a problem solving plan and perform it to change the real world and solve the problem.

On the other hand, learning processes get a version of a learning plan and perform it to build up understanding on the target world in their thinking world.

We divide learners’ work into two levels, called a meta level and an object level. The object level is mainly composed of problem solving and learning processes. The meta level is mainly composed of monitoring and planning processes.

The term “meta” is used because the monitoring/planning processes controls the learning processes and problem solving processes, that is, the monitoring process monitors and assesses (a) the progression of making problem solving plan, (b) problem solving effects and its progress, (c) the progression of making a learning plan, (d) the knowledge states and learning processes, (e) consistency between the problem solving plan and learning plan and (f) consistency between their own knowledge and observation result of problem solving behavior in the real world. More detail explanation will be given in Section 4.

The planning process revises problem solving and learning plans based on the monitoring and assessment results.

In general, learners make problem solving plans by iterating spirally those processes. Moreover, most learners tend to work ad hoc without explicit awareness of meaning, goals and roles of their activities.

The Web resources from our scenario are not illustrated in the figure, though all the processes refer them. This means that knowledge acquisition processes are fired in all the processes. It suggests that learners may forget the current question and deviate to another way depending on the contents of the web resources.

Our research goal is, therefore, to prompt to construct the rational spiral towards making/performing an effective problem solving processes by giving significant directions based on the task analysis.

3.2. A framework for modeling problem solving workflow

Based on the task analysis in section 2, we propose a framework where learners can make effective and efficient plans with explicit awareness of the structures of their work and distinction among their activities.

Hereafter, by problem solving workflow we mean all the required knowledge for problem solving and its flow, i.e., meaning and roles of target concepts, learning activities, problem solving procedures and so on.

3.1 Requirements

In this section, we examine the requirements for a useful framework for learners to plan their problem solving workflow.

The designed framework must
(i) support learners to represent learning processes,
(ii) support learners to get aware of the purpose of their work at the meta level,
(iii) support learners to make good use of web resources,
(iv) have a capability to check the consistency between problem solving processes and learning processes and,
(v) support learners to create highly sharable and reusable models.
Next, we explain why our framework must satisfy each of the above requirements.

For (i), learners tend to be more aware of problem solving processes than of learning processes. We, therefore, provide a framework to make effectively a plan of efficient learning processes.

For (ii), learners tend to be more aware of the object level processes, i.e., problem solving processes and learning processes, rather than the meta level ones. We, therefore, provide a framework that makes them have an explicit awareness of monitoring and assessing their understanding states of the target domain and progression of the planning. It suggests action that should be performed at the meta level to represent conditions of the target problem, learning processes for understanding the design rationales of the problem solving processes and reasons of version changing of the model.

3.2 Construction of a problem solving workflow

Figure 3 shows our proposed framework for representing problem solving workflow. It is composed of a knowledge map and a task flow model.

The knowledge map is divided into a knowledge resource and domain map. The domain map is considered as

![Figure 3. A model of problem solving workflow](image)

...
learner’s understanding of each concept, e.g. what is DOM is ‘known’. Thus, D-Map is considered to be a user model of the target domain.

The task flow is a model of a series of learner’s activities along time axis (each activity is expressed as a combination of a verb and a noun in natural language). The task flow model is divided into a problem solving process model and a learning process model. We, therefore, can consider them as an externalized representation of the PS Plan and LP Plan in Fig. 2., respectively.

- **Problem Solving Process Model (PSPM)** is a model of problem solving processes, a series of problem solving activities, on the target problem, for example, “install JBuilder,” “choose DOM or SAX system,” and so on.

- **Learning Process Model (LPM)** is a model of learning processes, a series of learning activities, for example, “gather information on DOM and SAX system,” “understand the characteristics of DOM and SAX,” and so on. It is recognized as a problem solving in acquiring the knowledge for performing the task.

The difference between PSPM and LPM is in the type of their effects. Activities in LPM affect learners’ states of understanding while those in PSPM affect the real world and contribute directly to solve the problem.

We have three reasons to prepare both LPM and PSPM and separately define them. The first is that the existence of the models gives learners’ explicit awareness of their task structures. It prevents learners from losing their way by getting aware of the purpose of what they do and what they should do next.

The second is shareability and reusability of models. PSPM can be widely shared and reused in performing similar tasks. LPM, however, may not be reused or shared even in performing similar tasks, since their activities depend on the learners’ expertise and states of understanding of the target domain concepts. Of course, learners whose states of understanding are similar to this of the creator of the model can share and reuse the LPM.

The third is that not only problem solving processes but also design rationales behind them are shared effectively by successive learning activities.

### 4. Design of an ontology for enhancing learners’ meta cognition of their work

As a result of software engineering research, we know that modeling methodology mainly includes product models and process models [4]. The product model specifies what constraints the model should satisfy and the process model specifies how it should be constructed. In the ontological engineering field, product and process ontologies, which correspond to the product and process models respectively, are developed in a machine readable manner.

We have introduced a product ontology and process ontology in the framework. The product ontology supplies basic domain-independent concepts and relationships for learners to represent the model and the constraints that should be satisfied by the model while the process one supports the learners’ modeling processes.

Product ontology supplies the concepts of process and object and the semantic links of “is-a” and “part-of” for the learners to represent the model. Learners make domain/task ontologies of the target world using them, and the consistency between the models are maintained by the system. For example, if they represent the learning processes for acquiring the knowledge for performing the problem solving processes, the constraint that the learning processes must be performed prior to performing the problem solving processes, e.g., learning processes for acquiring the knowledge on “DOM or SAX system” should be performed prior to the problem solving processes of developing an XML document retrieval module. The consistency are checked by the system. We call the pair of the processes in LPM and PSPM that holds above relationship as corresponding processes. It is justified by checking the processes share the same concepts in D-Map.

Meta cognition is a concept that includes both the meta cognition knowledge, i.e., knowledge about humans’ cognition activities, and the meta cognition control, i.e., a control process of cognition activities [6].

![Figure 4. The interaction makes planning processes](image-url)
(1) For the context of making a problem solving plan
- ((asses the validity of the problem solving plan)
  (choose-actions (gather kr)
   (re-planning pspp)
   (re-planning lp)
   (…))
  )
- ((Check the consistency between the results of problem solving processes and own understanding of them)
  (choose-actions (modifty d-map)
   (…))
  )
- ((Monitor the progression of performing problem solving processes and learning processes by referring each other)
  (choose-actions (re-planning pp)
   (re-planning lp)
   (…))
  )
- ((Check the consistency between the results of problem solving processes and own understanding of them)
  (choose-actions (modify d-map)
   (…))
  )
...  

(2) For the context of performing problem solving processes
- ((Monitor the progression of performing the problem solving processes)
  (choose-actions (re-planning pp)
   (…))
  )
- ((Check the feasibility of problem solving processes)
  (choose-actions (investigate the cause)
   (re-planning pp)
   (adopt alternative-pspp)
   (…))
  )
- ((Check whether the desired results/effects are obtained by performing the problem solving processes)
  (choose-actions (modify d-map)
   (…))
  )
...  

(3) For the context of making a learning plan
- ((asses the validity of the learning processes)
  (choose-actions (gather kr)
   (re-planning lp)
   (…))
  )
- ((Monitor the progression of performing problem solving processes and learning processes by referring each other)
  (choose-actions (re-planning pspp)
   (re-planning lp)
   (…))
  )
- ((Check the consistency between the results of problem solving processes and own understanding of them)
  (choose-actions (modify d-map)
   (…))
  )
...  

(4) For the context of performing learning processes
- ((Monitor the progression of performing the learning processes)
  (choose-actions (re-planning lp)
   (…))
  )
- ((Check the feasibility of the learning processes)
  (choose-actions (…))
  )
- ((Check the consistency between the results of problem solving processes and own understanding of them)
  (choose-actions (…))
  )
- ((Asses the validity of the learning processes)
  (choose-actions (…))
  )
...  

kr: knowledge resource
pspp: problem solving process plan
psps: problem solving process
lp: learning process
lps: learning processes plan
d-map: domain map

Figure 5. An ontology for enhancing the interaction loops between meta cognition and cognition activities

Figure 4 shows that the plan is detailed gradually along the time axis. In Figure 4 (a) shows a planning process when learners have the explicit awareness of the iteration loops and iterate the loops of meta cognition activities and cognition activities spirally, while (b) shows a planning processes with implicit awareness of them. It is not strange idea that the former case can make effective and efficient plans of problem solving workflow rapidly better than the latter. Learners tend to fall into confusion and lose their way without explicit awareness of the interaction loops, since the nested structures of their work and new information of the target world put heavy loads on them.

In our reference model, meta cognition activities are in monitoring/planning processes and cognition activities are in performing problem solving and learning processes. The interaction loops between learners’ activities are, in this case, the interaction loops of monitoring/planning processes and problem solving/learning processes.

Our research goal is to provide a support system where learners can make effective and efficient problem solving workflow in a poorly experienced domain by enhancing their explicit awareness of the interaction loops.

Needless to say, there is no general methodology to encourage the interaction loops independent of the target domain. However, the adequate stimulations appropriate to the situation have good effects on encouraging the interaction loops. Our fundamental approach to encourage the interaction loops is to provide learners with an actions list which should be performed at the meta level to get their explicit awareness of the interaction loops based on the process ontology. In the following, we illustrate the process ontology to encourage the interactions.

In the interaction loops, planning processes are basically triggered by the monitoring/assessment results and according to the results learners perform planning processes to dissolve them. We, therefore, provide a process ontology where each definition is composed of a pair of actions, i.e., an action to be checked for the current situation and an action list to dissolve the monitored/assessed erroneous parts.

Figure 5 shows process ontology for enhancing the interaction loops. The definition is described as follows.
and a task list. Learners can make reasonable problem solving plans and compose a check item list to learn processes. The lower right window displays a list of actions that they should take when something erroneous plans are observed.

For example, the first one in (1) defines that learners assess the validity of the problem solving plan and if they find out something invalid, they should gather knowledge resources and/or re-plan the problem solving processes and/or re-plan the learning processes, and so on.

In the figure, definitions are classified by the learners’ working context based on the task analysis. The definitions are for the situation (1) to make a problem solving plan, (2) to perform problem solving processes, (3) to make a learning plan and (4) to perform learning processes.

The items in (1) enhance the awareness of the interaction loops along (a) and (e) arrows in the reference model in Fig. 2 while the ones in (3) do the awareness along (c) and (e) arrows. The items in (2) enhance the awareness of the interaction loops along (b) and (f) arrows while the ones in (4) do along (d) and (e) arrows.

By providing an ontology suitable for the situation, learners can have an explicit awareness what they should do as meta cognition activities in their working context to make a consistent/reasonable plan and what actions they should take if erroneous parts are monitored. Thus, they can have a clear view of the purposes of their work.

5. **Kassist: A workbench for planning problem solving workflow**

It is quite time consuming work for learners to make effectively reasonable plans of problem solving workflow. To lighten the learners’ work, we are developing Kassist, an environment for planning, performing, sharing and evolving problem solving workflows.

The basic philosophy underlying the system is to support similar problem solving and learning by enhancing the meta cognition of learners’ work because most learners have weak awareness of them.

Of course, it is difficult for the system to capture the processes of the iteration loops in detail. The system, however, can give typical patterns of planning as meta cognition activities in their working context to make a consistent/reasonable plan and what actions they should take if erroneous parts are monitored. Thus, they can have a clear view of the purposes of their work.

Fig. 6 shows a designed screen image of Kassist. The upper left, upper right, and lower left windows display the editors for problem solving processes, domain map and learning processes. The lower right window displays a planning workspace that is composed of a check item list and a task list. Learners can make reasonable problem solving workflow with the functional support by the system.

In the domain map editor, they can model the concepts and relationships among them, in other words, their state of understanding of the target area is modeled and visualized in the pane. It encourages their awareness of their own state of understanding and (re-)planning learning processes as well as problem solving processes.

Furthermore, learners can link the concepts and relationships in D-Map to the learning and problem solving processes. This enables them to understand their contents and the corresponding relations among learning and problem solving processes. In the figure, if the learner clicks on “DOM system” with the right mouse button and then chooses “View LP and PSP” from the context menu, then the corresponding processes in the learning processes and problem solving processes are highlighted, in this case, “learn DOM system”, and “develop XML document retrieval module”. In addition, “JAXP” that includes “DOM system” and “learn JAXP” are highlighted in the editor. This functionality is realized by tracing the semantic links described in the model.

In the learning process editor and problem solving process editor, learners can drill down the processes to make more detail plans and refer the corresponding processes and the domain map. This function can smoothly switch the view of the model and check the consistency among models. Furthermore, learners can link useful knowledge resources to the process. Since the learning processes and problem solving processes capture the context, learners can assess the knowledge resources in the context appropriately and retrieve/reuse them effectively.

In the check items list of planning workplace, check items that learners should check in their working context are listed based on the process ontology. For example, when the learners perform problem solving processes, they have “Check the feasibility of problem solving processes,” “Check whether the desired effects are obtained by performing the problem solving processes” and so on. By checking the lists, they can make a valid plan and construct interaction loops based on the process ontology.

Consider the situation that the learner performs a problem solving process. He/she can be aware what they should do by referring the check item window and checking the feasibility of a process according to the check items. If he/she finds that the process is not feasible on account of some reasons, he/she can click the check item or the process. In the figure, if the learner clicks and chooses “Re-planning” from the context menu, then reasons for the re-planning corresponding to the checked items are displayed and then he can choose more detail reason, e.g. “because the method violates the constraints of the problem.” This context menu is displayed according to the definition of the process ontology. Using the functionality, learners can get aware explicitly of the meaning of their work and this encourages the interaction loops of their work.

In addition, the system has a function of version trace, workflow management and so on. We designed them based
on the task analysis to support the learners’ reflection and awareness processes of making learning and problem solving plans. This function allows learners to trace/check the planning processes and see the intentions of changes among version.

6. Concluding Remarks

In this paper, we analyzed the novice learners’ work and proposed a reference model of their work. Based on the task analysis, we designed a framework for representing problem solving workflow and process ontology to encourage learners’ awareness of their work. Furthermore, we designed the system Kassit that supports novice learners planning processes of problem solving workflow.

We can view D-Map as a valid domain ontology [3] sufficient for performing the problem solving processes. Thus users and the system can collaboratively retrieve useful information from the WWW using the semantic information. Furthermore, we understand PSPM, LPM and D-Map play a role of annotation of knowledge resources. PSPM and LPM capture the context of using knowledge resources, so we might manage them effectively by using these information. This subject will be addressed more carefully in our future work.

References


Abstract

Existing approaches to ontological support in courseware are focused on the annotation of hypermedia nodes or contents, while links are described only by classifying them into a set of predefined types. In this work, we describe the concept of Learning-Link as an independent and reusable learning resource that can be annotated with terms defined in a link ontology to specify its type. The computational semantics of that ontology, combined with fuzziness to specify graded relations between pairs of link sources and targets for specific users or groups, enable the implementation of extended adaptive behaviors that may entail inferences and approximate reasoning. A prototype implementation of Learning-Links on a SCORM-compliant runtime environment, based on XLink definitions, is described as a proof of concept.

1. Introduction

Existing approaches to using ontologies (or similar knowledge representation formalisms) in learning resources are mainly focused on mapping concepts to hypermedia nodes or contents, while links are usually constrained to a number of predefined interpretations, which are in some cases derived from the meta-structure of nodes. For example, in MYENGLISHTEACHER [1] the system derives connections of two types (priority and relatedness) from terms that are used to annotate contents, and in ELM-ART [2], concepts in a network are related to nodes.

In addition, current proposed standards for the interoperability and reusability of learning resources (and specially, the SCORM model [4], that aims at being a unified reference model) are organized around the concept of learning object (Sharable Content Object, SCO, in SCORM), defined as structured, context-independent aggregations of Web assets. In consequence, the model is also oriented towards metadata annotation of contents, neglecting link annotation. In fact, in SCORM, the sequencing links that conform the navigation scheme are derived implicitly from content structures [3], and other kinds of explicit links are limited to internal sequencing inside learning objects, as described in its run-time specification [12].

Nonetheless, ontologies can be used also as an extension of existing link-type listings like [10] or [14], or as an enhancement of extensible link typing systems, like that of NOTECARDS [18]. This way, link ontologies can be used to annotate the links themselves, so that richer and unified models of linking can be exploited to provide improved interaction visualizations [17], and to design adaptive behaviors, as sketched in [3]. In order to achieve these goals, the most salient features of current ontology languages oriented to Web technologies [21] are reasoning around subsumption (i.e. about categories and subcategories defined by terms, see [15]) and the consensual aspect of shared ontology development [22], which promises the availability of commonly agreed terminologies. In addition, research has pointed out that link types are related to some of the usability properties of the hypertext structure [13], so that its systematic classification can be considered an important factor in the design of adaptive linking [19] technologies.

In this work, we describe the novel concept of learning link as a proposed first-class citizen in educational technology reference models, and how XLINK1 and a number of straightforward extensions to the SCORM model can be used to describe learning links. These links can be reused independently of the contents they associate, and ontologies can be used to provide them with semantically rich types. In addition, imprecision in the implicit assertions contained in links is considered a key characteristic that may enhance the implementation of adaptive behaviors that are supported by learning links.

The rest of this paper is structured as follows. In section 2, the concept of learning link is described as a context-independent, typed entity that can be used to

1 http://www.w3.org/TR/xlink/
represent (possibly imprecise) semantic relationships between learning resources. Section 3 describes how this concept can be mapped to the content model of the current SCORM standard, and illustrates it through a prototype implementation in which links are defined separately from content elements. Finally, conclusions and future work are provided in section 4.

2. A Learning Link Model Supporting Types and Fuzziness

The annotation of links with ontology terms is a generalization of the concept of link type [5] that has been used in hypermedia models [6] for diverse purposes. Web-enabled ontology definition languages [7] enhance simple link types with the possibility of defining arbitrary relationship definitions between types and with full expressive power for the definition of axioms that can be used for subsequent reasoning processes (since they are based on description logics [15]). In consequence, applications that rely on link typing [8] may take advantage of these extensions.

In addition, vagueness is an inherent property of many link types, since they convey semantic structures that are essentially vague. For example, weights in [1] can be considered an implicit form of quantifying grades in relatedness connections. The concept of fuzzy link [9] can be used to model that vagueness depending on the learner and the semantics of the link type. As a result, a link can be defined in an abstract manner as a triple \( L=(S, T, \Lambda) \), where \( S \) and \( T \) are the sets of sources and targets, respectively, and \( \Lambda \) is a set of membership functions in the form:

\[
\mu_{L,p}:(S,T,U)\rightarrow[0..1]
\]

where \( p \) denotes the type or purpose of the link, and \( U \) is the set of users of the system. An arc that belongs to the link \( L \) in the form \((a, b)\in S\times T\) with a specific type is thus described by the corresponding membership function, which allows the definition of partial matching with specific users (this is essentially a generalization of the abstract link model of XLINK). For example, the relative density level of a set of nodes \( D \) that give details about a node \( k \) in \( S \) along with the mastery level of the user (in the corresponding concept represented in \( k \)) can be used to derive a membership grade in a link so that:

\[
\mu_{L,\text{density}}(k, d, u) = f(\text{density}(d), \text{mastery}(u)), d \in D
\]

where \( f \) is a function that should be empirically adjusted.

If we use a term defined in a link-ontology as a link type designator, we can take advantage of ontology-defined predicates. For example, we know that an analogy link is a kind of argument link, which is related to a work-part according to [10]. Furthermore, designer-defined links can be specified on existing ontologies, for example, the “where in the world?” link cited in [8] could be considered a subtype of apply link in [10].

3. Extending SCORM for Learning Links

The current SCORM content model [4] is organized around the abstract notion of “learning resource”. Currently, three specializations of this notion are included in the proposed standard: assets (representing any form of electronic media), SCOs (representing collection of assets that conform a reusable, subjectively small unit of instruction), and content aggregations (that define the structure and sequencing of assets and SCOs). The static relationships between these three kinds of resources are depicted in Figure 1 as a UML diagram. An SCO can be made up of a number of assets – with one of them being a “launchable asset” that initiates and maintains communication with the Learning Management System (LMS) –, and can also be an aggregation of a number of other SCOs. Content aggregations define the structure of SCOs and assets, indicating possible sequencing strategies to the LMS.

![Figure 1. The current SCORM Content Model](image)

3.1. Description of Learning Links in SCORM

We’ve specified a fourth type of SCORM learning resource defined to introduce learning links as sharable elements, as a new entity called SLO (shareable link object). It should be noted that the SCORM specification is deliberately open for extension in the kinds of learning resources, as expressed in section 2.1.1 of [4]: “ADL envisions more specializations of learning resources to be introduced in future versions”.

Independent links between SCOs – or link-bases expressed, for example, in XLINK format –, can be described with the same metadata items used for the other kinds of resources, thus not breaking the current restriction which requires that “learning resource sequencing is defined in the content structure and is external to the learning resource”[4]. Since SLOs, like SCOs, are reusable and context-independent, they can be considered to be of the same status of SCOs in that they
are tracked by the LMS, but they possess different semantics [12] in that they are able of launching other SCOs, and in that more than one SLO may be active at the same time.

The annotation of the links should be defined on metadata items rather than in the link itself, to be LMS-visible. Basically, the meta-metadata section should include references to the ontology, and the classification section can be used to annotate the link (or set of links) with specific classes.

3.2. Implementation Case Study

As a proof of concept, we have developed a link ontology derived from Trigg’s taxonomy of scientific writing [10], and a preliminary prototype based on the reference implementation provided by the ADL Initiative, called “SCORM sample run-time environment 1.2”\(^2\). The prototype uses the DAML+OIL RDF-based markup generated by the ontology editor called OILEd\(^3\) (a screenshot of a part of the ontology is showed in Figure 2).

Implementations can use the DAML+OIL markup generated (which in turn is RDF(s)-based), in which each term is identified by a URI. For example, the Analogy Link type is described by the following markup:

```
<daml:Class rdf:about="http://www.dei.uc3m.es/Trigg.daml#ArgumentLink-Analogy">
  <rdfs:label>ArgumentLink-Analogy</rdfs:label>
  <rdfs:subClassOf>
    <daml:Class rdf:about="http://www.dei.uc3m.es/Trigg.daml"/>
  </rdfs:subClassOf>
</daml:Class>
```

A SLO in a SCORM manifest describes a set of arcs (sources and targets) between assets or SCOs. In consequence, it represents a different link concept to the one represented by “internal links” (i.e. common HTML links). The latter can be embedded in learning resources, but are not allowed to link resources outside their embedding context [12] according to SCORM.

A typical imsmanifest.xml file describing an instructional unit is made up of three major sections as depicted in Figure 3. The meta-data section describes the elements that are referenced in the resources section as a whole, and the optional organizations section defines content structure and sequence.

For our purposes, we have to include a new kind of element in the “resources” section representing learning links (SLOs), which in turn, according to SCORM best practices, are described in a separate meta-data file. For example, the following XML fragment describes a SLO:

```
<resource identifier="slo_1" type="xlink" adlcp:scormtype="slo" href="slo01.xml">
  <metadata>
    <schema>ADL SCORM_LL_ext</schema>
    <schemaversion>1.2</schemaversion>
    <adlcp:location>
      slo01_meta.xml
    </adlcp:location>
  </metadata>
  <file href="slo01.xml"/>
  <dependency identifierref="sc03"/>
  <dependency identifierref="sc04"/>
  <dependency identifierref="sc05"/>
</resource>
```

The non-standard elements in the SLO definitions are the following:

- The type attribute contains the string “xlink” instead of the generic “webcontent” value.
- The scormtype attribute identifies the resource as a SLO. This modification requires a change of the XML Schema definitions of the standard if validation is required.
- The name of the schema is changed.
- The SCOs that are linked by the SLO are specified as dependencies. This information is redundant with the definition of the SLO itself, but is used for consistency with current SCORM packaging practices.

\(^2\) Available at http://www.adlnet.org/

\(^3\) Available at http://oiled.man.ac.uk/
Figure 3. Description of a SCORM and the place of SLOs

The file containing the metadata of the SLO (slo01_meta.xml following the above example) would include a reference to the ontology along with one or several of its terms that act as link types, as illustrated in the following markup fragment:

```
<metametadata>
  <metadatascheme>ADL SCORM 1.2</metadatascheme>
  <language>en-US</language>
  <metadatascheme>
    http://www.dei.inf.uc3m.es/Trigg/
  </metadatascheme>
</metametadata>
<classification>
  <taxonpath>
    <source><langstring>
      http://www.dei.inf.uc3m.es/Trigg/
    </langstring></source>
    <taxon>
      <entry>AnalogyLink</entry>
    </taxon>
  </taxonpath>
</classification>
```

Then, the definition of the arcs of the link is contained in the file that contains the XLINK definition (slo01.xml). Following our example, the following markup defines a link from a concept page (labeled origin) to its related explanations (labeled target):

```
<?xml version="1.0"?>
<slo
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="extended">
  <loc xlink:type="locator"
    xlink:href="/sco03" xlink:label="origin"/>
  <loc xlink:type="locator"
    xlink:href="/sco04" xlink:label="target1"/>
  <loc xlink:type="locator"
    xlink:href="/sco05.htm/h1"
    xlink:label="target2"/>
  <arc xlink:type="arc" grade="1"
    xlink:from="origin" xlink:to="target1"/>
  <arc xlink:type="arc" grade="0.7"
    xlink:from="origin" xlink:to="target2"/>
</slo>
```

When defining locators, an XPATH location can be optionally specified, so that links can be applied to elements of lower granularity than SCOs. A similar approach is used by the annotation tool called ANNOTEA [16].

Membership grades can be specified in the meta-data, as hard-coded values or functions or they can be internal to the LMS. In the above link, a grade of interconnectedness has been specified from the source to each of its targets, by using a grade tag and numerical values normalized in the [0..1] interval (linguistic labels like ‘very high’ or ‘rather low’ representing predefined fuzzy numbers could have been used instead). These grades are not related to the user model, of the application, but could be combined with it.

To implement a fuzzy link $\mu_{L,analogy}$ from the given definitions the LMS must internally implement a membership function. For example, we can use the following simple membership function (for illustration purposes):

$$\mu_{L,analogy}(s, t, u) = \text{grade}(s,t) \cdot (g(\text{cmi.core.lesson_status}))$$

where $\text{grade}(s,t)$ denotes the grade of the corresponding arc, and the function $g$ takes the value of the SCORM lesson status of the SCO and:

- if the status is not_attempted, browsed, incomplete or failed, it returns one. This way, the compatibility for the user is that of the arc.
- if the status is completed, it returns 0.5, since the user is known to have experienced all the elements in the SCO, and thus, detailed analogies are not considered so relevant.
- If the status is passed, the score of the user (normalized in the [0..1] interval) is returned. Higher scores represent lower relevance of the analogies.

Adaptation can be built on top of fuzzy links. For example, we can change the size of the font of the link by an abstract rule like the following:

```
if L.type is ArgumentLink and L
then L.font-size is big
  and L.font-family:=Verdana
```

In the rule, the first antecedent checks if the type of the link is argument (or any of its subsumed classes, like AnalogyLink). The second antecedent is simply substituted by the compatibility of the link with the current user, and would yield a fuzzy value. In consequence, the combination of both antecedents is a partial truth-value. In turn, the first consequent assigns a (fuzzy) size to the font-size of the link for the current user, while the second one simply assigns it a specific font-family. Rule activation can be implemented with existing well-known semantics, and common defuzzification methods can be used to
obtain a crisp size from the fuzzy inference. We have used the ones provided by the Fuzzy Java Toolkit (FJT) libraries (http://www.iit.nrc.ca/IR_publication/fuzzy).

The class LMSManifestHandler was extended to read the extended manifest syntax, and storing learning links in the relational database (using Fujitsu XLIP – http://www.labs.fujitsu.com/free/xlip/en/) libraries to parse XLINK definitions). After that, the JSP script file sequencingEngine was modified to add LMS-controlled links on the fly, depending on the current knowledge status of the learner, described by the SCORM data model. For example, detail links (and links of any of its subclasses) may only be showed by default if cmi.core.lesson_status is 'passed', and the inverse may occur with summarization links. Extensions to the cmi.student_preference data item can be used also for adaptation (for example, to hide or show formalize links depending on the student's mathematical background). In addition, membership values can be used also as sorting or highlighting criteria in adaptive link sorting or annotation.

Figure 4 shows the appearance of the modified SCORM environment. Since SCOs are context-independent content elements, the inclusion of learning links in the page had to be done at the browser level, dynamically manipulating the document object model of the SCO through server-generated javascript functions. Concretely, we have used a frame (normally hidden), with a shaded background in the figure, that is reloaded on every navigation to check for links that apply to the current SCO and user, and subsequently updating the SCO. In the Figure, a highlighted link produced from one of the above-described XLINK definitions is shown. This approach enables the implementation of adaptive linking technologies on SCOs and SLO that are designed independently. Links are added at the end of the SCO by default, or they are put in specific parts of the SCO if a specific XPATH location was specified in the description of the link.

Generated links provoke the activation of a LMS behavior that launches the linked SCO. Multi-target activations are for now precluded since they break the SCORM requirement that SCOs must be launched one at a time.

In the just described example, we have only illustrated one of the multiple applications of link types in instructional design. But the same technological support can be used to design other specific uses of link types in hypertextual learning structures like those described in [20].

4. Conclusions and Future Work

A model of learning links as reusable learning resources has been described, with support for vagueness and semantic typing. Our current approach can be situated at level-2 according with the computational semantics described in [11] and complements existing approaches that use ontologies or concept maps to provide semantic interpretation to hypermedia nodes or contents. The link
model can be mapped to XLINK structures and integrated in existing learning technology standards.

Future work will be carried out in how (approximate) sequencing strategies can be derived from concrete link types. In addition, an open avenue for research is open on how description logics-based reasoning processes can be used to assist in link adaptation.

References

Authoring Framework for Concept-based Web Information Systems

Lora Aroyo¹ and Darina Dicheva²

¹Technische Universiteit Eindhoven, The Netherlands
l.m.aroyo@tue.nl

²Winston-Salem State University, United States of America
dichevad@wssu.edu

Abstract

Authoring of concept-based WIS aimed at providing adaptation or various sorts of personalization to the user needs intelligent support. An approach to such support with regard to both application authors and system developers should provide a link between high-level author’s tasks and low-level developer’s functions. We propose an ontological (in compliance with the Semantic Web technologies) approach towards a common authoring framework to formally describe the process of presentation generation in concept-based WIS. A key idea is to separate completely the content and the application related issues within WIS, which will allow for better modeling of the application-related processes and for making accessible and retrievable external data in adaptaive WIS.

1. Introduction

WWW grew fast from being just a distributed hypermedia system with basic navigation capabilities to a collection of distinct information systems applications, using various information retrieval approaches. Nowadays it becomes a crucial necessity to find common ways of accessing information on the Web and also of constructing data-driven Web Information Systems (WIS) maintaining it. A great deal of current research efforts focuses on WIS content development and reusability as well as on the authoring process of content access and presentation.

The main goal of a WIS is to automatically generate hypermedia presentations based on the system data (content, which is inside or outside of the application) and metadata in a specific application domain. In this process some WIS take in consideration the current user’s preferences, goals and knowledge and attempt to automatically generate an adaptive (user-oriented) presentation. There are various presentation/navigation strategies that could influence the final navigation model of the information presentation. These could be, for example, adaptive hypertext [5], instructional [3] or rhetorical [9] information perception strategies. The adaptivity in those cases depends on the specification of the domain model (application area) and user model (user’s goals, activities, and desires) [6,3]. The user model enables system adaptation to the user’s activities while interacting with it (for example browsing or searching for content).

It is recognized that the Web would benefit from explicit semantics for its content and that ontologies will play a key role in providing infrastructure to support it. This leads to increasing attention to conceptual modeling and concept-based WIS. The Semantic Web initiative does a big effort in this direction. Semantic Web technologies allow for WWW data to be more machine understandable and in this way enable the design of WIS that can reason over and automatically support the processes of authoring both the content and the model of content delivery.

WIS applications design and development currently involve the traditional handcrafted authoring approach [10]. However, as it is constantly more difficult for existing WIS to cope with Web information explosion and maintain and update their data timely, there is an increasing demand for more structured approach towards WIS content and process authoring. In addition, there is a recent attention to generating adaptive (user-oriented) information presentations [5,4,9]. This aspect of personalisation and adaptation in addition to the ones of content development and reuse complicate a lot the authoring process. Therefore an integrated and more semantically oriented approach is needed in order to facilitate the different stages of this process and help both authors and interface developers perform their tasks. A structured framework would facilitate the authoring by allowing authors and interface developers specify their goals and decisions and by supporting controlled checks (reasoning) at different stages of the engineering process (e.g. requirements analysis, design etc.) as well as from different perspectives of the application domain (e.g. with respect to content data, its structuring, sequencing, navigation, personalisation, adaptation and user presentation).

In this paper we propose an ontological (in compliance with the Semantic Web technologies) approach towards a common authoring framework to formally describe the process of both content and presentation generation of concept-based WIS. Such an approach is appropriate for systems like AIMS and AHA! with a clear distinction between the domain knowledge and the task (in this case instructional) knowledge, as this allows for clear separation of the concerns with respect to the authoring process as well. We begin with a discussion on authoring of WIS. Further, we propose an ontological approach to concept-based WIS authoring, exemplified in
two existing systems – AIMS [3] and AHA! [5,6]. Finally, we present an example and some conclusions.

2. Concept-based WIS and Authoring

Concept-based WIS have as a goal to generate automatically hypermedia presentations of the content in a selected application domain. This is usually done by employing a conceptual domain representation in order to link information items (the content) to the presentation (sequencing) structure. Therefore the concepts are the core building blocks underlying the main operations in the along going authoring for both content and process. A main characteristic of WIS is that they retrieve data on users’ request and provide the users with a hypermedia presentation of the query results within a specific context. In this process WIS may take in consideration the current user’s preferences, goals and knowledge represented in a user model (UM) and attempt to automatically generate an adaptive (user-oriented) presentation. An essential characteristic of this approach for generation of user-oriented information presentations is further separation of the user knowledge and the navigating strategies from the already separated conceptual (domain) knowledge and informational space (WIS content).

A concept-based WIS typically has a layered information structure, including:
- information object (Web document) repository,
- ontological (semantic) layer, conceptually modelling an application domain, which also connects domain concepts to repository objects or external Web documents,
- a user model and/or other application-related models used to generate a presentation of the data on user’s request.

Our approach is to completely separate the content from the application by introducing three layers: a content layer, an application layer, and a presentation layer. This allows for designing a kit of different applications using the same ‘content’. This scheme works well in both cases: WIS containing the data, which is to process (see Fig. 1) and WIS using external data (see Fig. 2). The content layer contains only the data (information objects) and the conceptual model (ontology) for this data. The application layer contains all the application related models. On the first place it includes a task model determining the main application task, e.g. a course model in instructional systems as AIMS, a simulation model, or an adaptation model, as in AHA. We consider the separation of the task model from the domain model very important, since in systems where these two are combined the complexity is increased dramatically and as a result the systems are either too complex, or oversimplified. It also includes a UM and possibly other application related models. It is important that the application layer includes also a domain model, which defines the link between the ontology (in the content layer) and the application in order to facilitate the retrieval of the actual data. The presentation layer contains the presentation model, which is responsible for the final generation of the information presentation (based on the user model and other application related models).

This architecture is a step further from the information model within AIMS [3], which includes a domain model (directly constructed over the data model), course model (an example of a task model describing how this data should be used by the user with respect to the course tasks), library model (defining an index to access the data) and user model stirring the navigation and the information retrieval mechanism.

![Figure 1. Concept-based WIS containing the content within the application](image)
By introducing the above layers we achieve clear separation of the content, the actual application, and the presentation, which is a crucial element in describing further the authoring activities in WIS. This is especially important with respect to complexity of the engineering processes involved. In this way we can do the authoring at the three layers separately, i.e. to introduce three layers of authoring tasks - content, application and presentation.

Content authoring in our scenario concerns creation of information objects such as Web documents and their annotation (e.g. creation of metadata, marking-up), and involves creation of links (conceptual and functional) between information objects. Although Web document (information object) authoring is a very important part of a Web-based information system we will not focus on it. Such an authoring traditionally includes the use of editors (e.g. text, HTML, XHTML, etc. editors) as well as document conversion. These forms of authoring require authors to describe documents in a markup system and to possibly consult a DTD.

Application authoring concerns the construction of conceptual structures, i.e. application domain model, as well as the task model and a user model.

Process authoring (presentation generation process) concerns content (information) delivery to the user, e.g. patterns of browsing, automatic generation of presentations of (retrieved) data, etc. Thus it mainly involves the definition of strategies to be followed in navigation/information retrieval and sequencing of information objects, as well as for their presentation to the user. Currently a lot of stress is put on presenting the content adaptively and personalized considering the individual characteristics of the users and the context of their work.

These authoring tasks are very time and effort consuming and will become even more demanding with the constant increase of the information available on the Web and with the involvement of complex strategies for Web content presentation. In addition, concept-based information systems often lack good authoring interface and require low-level programming skills from the content experts. Thus, there is an increasing need for efficient support for the WIS designers and builders (authors). We envisage that such support should include automatic or semi-automatic performance of some authoring activities, intelligent assistance to the authors in the form of hints, recommendations, etc., as well as templates based on recognizing different information patterns within domain ontologies or presentation (sequence) patterns.

In the next section we propose an approach to facilitate these authoring tasks based on two milestones: enabling WIS with structured and conceptual knowledge about the authoring process and using this machine processable knowledge to provide the authors with more user-friendly and efficient tools for authoring. Thus, we aim at a formal specification of the authoring process, based on a definition of Authoring Task Ontology (see Fig. 3). Initially we focus on the authoring at the Application layer of the WIS architecture.

3. An Ontological Approach to Concept-based WIS Authoring

As we already mentioned our goal is to provide semantically enriched structure of the authoring process of
concept-based Web information systems, which can be used to support application authors (see Fig. 4). We take an ontological approach towards the activities involved in the authoring tasks. WIS Authoring Task Ontology translates the application-related operations within DM, TM and UM into semantically organized authoring tasks, used by the Operational and Assisting layers, which provide the actual support to the author.

![Figure 4. Authoring modules](image)

WIS Authoring Tasks Ontology (WATO) is functional concept ontology in the sense of [7,1], where the functional concepts are WIS authoring tasks. WATO design involves an initial decomposition of the WIS authoring process into a set of basic authoring operations. The ontology describes the basic authoring functions and their mapping to the underlying application-specific models (such as the ones used in AIMS, AHA! and other WIS).

We propose that WATO has three main layers: base layer, which includes a set of atomic authoring tasks, ontology-layer, which includes a hierarchy of authoring tasks classes formed by conceptual categories of relationships (interdependence) between the primitive functions, and a meta-layer, which includes a hierarchy of meta-functions supporting application-related relationships. The atomic authoring tasks are primitive functional concepts, which are basic for the concept-based WIS authoring process, for it’s understanding and performing. Formal definitions of the atomic tasks are presumed to support their interpretation. This will allow building WIS authoring ontology terminology (vocabulary).

The primitive functions are defined on objects (e.g. concepts, links, tasks) within a specific concept-based structure (domain model, task structure model). They express a simple functional formalism, where the object changes the structure, or the structure is manipulated. Examples of atomic authoring functions include: Create (Structure), Create (Object), Add (Object, Structure), Delete (Object, Structure), Edit (Object, Structure), Link (Object, Objects, Structure), Delete (Structure), Compare (Object, Objects), Exist (Object, Structure) and List (Objects, Structure), where Object ∈ \{DomainConcepts\} ∪ \{TasksItems\} ∪ \{UMAttributes\} and Structure ∈ \{DomainModel, TaskModel, UserModel\}. Note that such a definition is independent of the information structure - the only prerequisite for it is to be concept-based.

In the ontological layer we define a hierarchy of authoring classes related by 'sub-task-of' and 'peer-task-of' relationships (representing ‘part-of’ and ‘is-a’ relationships correspondingly):

- 'sub-task-of' represents specialization between two tasks (e.g., DefineDomainConcept is a subtask of the DefineDomainConceptMap task). We assume that, each task authoring collection has at most one super-community (e.g. DomainAuthoring is the super-community in this case).
- 'peer-task-of' relationships are viewed as a referral mechanism to other authoring collections, which are considered as peer related for various contexts. A weight (a real value between 0 and 1) might be attached to a ‘peer-task-of’ relationship to represent the degree of relevancy as a peer.

These relationships present certain aggregation criteria that are used for grouping primitive functional concepts into higher-level authoring functions (classes). This way we can construct/identify functional groups of authoring tasks.

In the meta-layer, we define a hierarchy of application-related meta-functions to represent categories of application-related relationships (including causal and other relations among components). The meta-functions may represent a role (application-dependent) of one base-function for another base-function [7]. They are concerned not with the actual change in the objects, but with their actual function in the concept-based WIS authoring process. The following are examples of relationships between tasks within the meta-layer:

- 'is-preceded-by', 'is-followed-by' represent temporal relationship between two tasks,
- 'follows-from', 'is-required-by' ('is-a-prerequisite-for'), 'requires' represent causal relationships between two tasks,
- 'is-assigned-to', 'is-achieved-by', 'is-delegated-to' represent task-agent relationships.

We plan to define the meta-functions with conditions for their primitive parameters in order to achieve specific authoring goals, e.g. if <condition-task1> then <action-task2>, where condition might express the status of a task, and action could be the necessity of planning/postponing completing/quitting etc. another task in that case. This will be based on identifying/extracting the functional structure for WIS authoring from existing authoring models (as domain) and their connection to the specific application information.

In order to be able to extract from WATO additional semantics for the authoring process, we need to define rules to query the ontology and find patterns and alternatives in the navigation within the network of authoring tasks. This
AHA! and in Section 6 we present an excerpt of the model authoring collections in the context of AIMS and framework. In the next two sections we illustrate domain common patterns and use them as a basis for our proposed AHA! – for provision of content adaptation, to extract provision of task-oriented information handling support and based Web systems, including AIMS – a system for analyzing the basic authoring activities in some concept-in the repository.

In order to define these authoring collections we are analyzing the basic authoring activities in some concept-based Web systems, including AIMS – a system for provision of task-oriented information handling support and AHA! – for provision of content adaptation, to extract common patterns and use them as a basis for our proposed framework. In the next two sections we illustrate domain model authoring collections in the context of AIMS and AHA! and in Section 6 we present an excerpt of the suggested WATO DM authoring collection.

4. Domain Authoring in AIMS

4.1. AIMS

AIMS (Agent-based Information Management System) aims at providing combined adaptive information support for students and instructors within the context of on-line course environments [3]. The main goal is to improve the usability and maintenance of information in such environments, thus the focus of the work is on information search, retrieval and effective presentation to the user. AIMS information model consists of subject domain, course, library, and user profile. This way we make a clear distinction between the domain description (domain model) and the task pre-requisites description (course model) in order to support their efficient handling. We envisage domain concepts as the main linking component for all the objects within these models.

The domain model defines subject-domain ontology and is represented as a concept map (CM) of domain concepts and links between them. The link types are based on the generic selection of link types defined in [8]. Domain concepts are linked to documents and each link is assigned a weight indicating document relevancy to the concept.

The course model defines the pre-requisite structure of a course and includes course topics and course tasks. Course tasks are pre-defined in a course task library and correspond to course assignments that the student is required to perform. Each course task is represented in terms of domain concepts, which the student must learn to successfully complete the task, and some additional information, such as task description, prerequisites, and task status. Course tasks are related to course topics; thus associating a topic with the collection of domain terms included in all related tasks.

The library is a collection of information items related to different courses and domains. Each document is described by its name, author, year of publication, location, short description, list of weighted keywords, presentation format, and instructional format. Library documents are related to domain terms and through them to the course tasks and topics.

The user model is an overlay model of DM and contains a set of values of the attributes for each DM concept. As a result, the process of authoring of such concept-based Web courseware includes domain-, course-, library and UM authoring. Domain authoring concerns constructing a domain concept mapping structure. It includes adding, deleting and updating 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 author can also create new types of links. Course authoring concerns defining the structure of a course within a specific domain. The author defines course topics and tasks and assigns a list of keywords to each task. Since the domain terms are linked to library documents, this provides a connection between the course structure and the appropriate course material. A topic usually corresponds to the course weekly session and the tasks to the course weekly assignments. Library authoring includes adding, deleting and updating library documents and their links to domain concepts. UM authoring includes updating the values of the concept attributes, adding and deleting concepts and attributes in UM.

4.2 Basic Domain Authoring Tasks in AIMS

In this section we present basic tasks (part of the base-layer, see Section 3) for domain authoring in concept-based courseware. These tasks are defined using a set of atomic operations, summarized in Table 1. Note that the same set is used for defining the other two authoring tasks collections in AIMS (course and library). The following abbreviations are used: CM = Domain Concept Map, DirLCo = Set of Directly Linked Concepts, RelC_o = Set of Related Concepts, RelD_o = Set of Related Documents.
Table 1. Atomic operation definitions (adapted from [2])

<table>
<thead>
<tr>
<th>Atomic operation</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘Add’</td>
<td>${T_o, T_x, C_o, L, D_o}$, where $T_o \in {\text{course topics}}, T_x \in {\text{course tasks}}, C_o \in {\text{domain concepts}}, L_i \in {\text{domain links}}, D_o \in {\text{library documents}}$.</td>
<td>Adds an object to the corresponding structure, i.e. $T_o, T_x, C_o$, etc.</td>
</tr>
<tr>
<td>‘Del’</td>
<td>As above</td>
<td>Deletes an object from the corresponding structure.</td>
</tr>
<tr>
<td>‘Edit’</td>
<td>As above</td>
<td>Edits an object settings.</td>
</tr>
<tr>
<td>‘U’</td>
<td>${CM, CS, EML}$, where $CM=\text{Concept Map}, CS=\text{Course structure}, EML=\text{Educational Metadata Library}$.</td>
<td>Updates the corresponding information structure, i.e. CM, CS or EML.</td>
</tr>
<tr>
<td>‘L’</td>
<td>${\text{DirLC}, \text{RelC}, \text{RelC}_o, \text{RelT}_o, \text{RelD}_o}$, where $\text{DirLC} = \text{Directly linked concepts}, \text{RelC} = \text{Related courses}, \text{RelC}_o = \text{Related concepts}, \text{RelT}_o = \text{Related tasks}, \text{RelD}_o = \text{Related documents}$.</td>
<td>Lists the objects of the set(s).</td>
</tr>
<tr>
<td>‘V’</td>
<td>${\text{Graph}, \text{Text}}$, where ‘Graph’ is a graphical and ‘Text’ a textual view.</td>
<td>Gives alternative views of the engine results to the author.</td>
</tr>
<tr>
<td>‘Chk’</td>
<td>${T_o, T_x, C_o, L, D_o, \text{RelC}_o, \text{RelT}_o, \text{RelD}_o, \text{DirLC}}$</td>
<td>Checks the existence of objects within set(s).</td>
</tr>
</tbody>
</table>

At the highest level of the AIMS domain authoring tasks hierarchy are tasks such as creating a domain, editing a domain, copying a domain, exporting a domain, importing a domain, comparing domains, merging domains. These tasks involve primitive concept-maintenance related tasks such as add/delete/edit a concept, create/delete/edit a link between concepts, create/delete/edit link type, etc. Table 2 presents an excerpt of AIMS primitive domain authoring tasks at this level together with associated actions of AIMS Domain Assistant and Domain Engine (see Fig. 4).

As a specific example let us consider the task ‘add concept’, i.e. Add (C, CM). A concept in AIMS is represented with the help of the following format:

```
<concept>

  <name> </name>
  <author> </author>
  <description> </description>
  <resource> </resource>
  <weight> </weight>
  <variants> </variants>
  <child> </child>
  <parent> </parent>
  <context> </context>
</concept>
```

Thus, this authoring task requires for the author to specify concept attributes; i.e. to carry out actions such as: specify concept name, specify concept description, indicate resources for this concept, indicate concept parent(s), specify concept’s synonyms. These can be achieved by performing the following subtasks:

- specify (fill-in) concept’s (object’s) attribute
- indicate concept’s relation to another concept.

This in turn includes the following subtasks:

- select a concept
- specify link’s attributes, e.g. type.

On the other side, ‘add concept’ is a subtask of the higher level tasks ‘create CM’ and ‘edit CM’. Other examples of domain-authoring tasks at a higher level include removing all direct links to a concept, removing all segments of a path between two concepts, or creating links between the domain structure and the library. Such tasks can be implemented with repetitive calls to atomic operations. For example, the composite tasks ‘delete all direct links of given concept’ can be implemented with a repetitive call to atomic operation ‘delete link in CM’.

5. Domain Authoring in AHA!

5.1 AHA!

AHA! (Adaptive Hypermedia Architecture) is a Web-based adaptive hypermedia system able to perform adaptation based on user’s browsing actions, regardless of the interpretation of browsing, e.g. as a learning process, exploration, etc. AHA! makes use of the AHAM reference model, which provides a general framework to describe the adaptation functionality of adaptive hypermedia systems (AHS). Similarly to the AIMS information model, AHAM also advocates clear separation of DM, UM and AM within the AHS design, which refer to the (1) information domain, (2) representation of the user and how s/he relates to the information content, and (3) description of how AHS adapts the information (presentation and navigation) to the user [6].

In the AHAM domain model, which is based on the Dexter reference model, a central notion is component (concepts and concepts relationships). A page is considered to be composed of several fragments, which are atomic components.
Table 2. Basic domain authoring tasks (adapted from [4])

<table>
<thead>
<tr>
<th>Task</th>
<th>Domain Assistant</th>
<th>Domain Engine</th>
<th>Result</th>
</tr>
</thead>
</table>
| Add (C, CM) | • suggest options for the author:  
  - add or delete domain engine results  
  - give alternative presentation:  
    - V (Text, ReC, Relevance %)  
    - V (Graph, Domain Ontology, Matched concepts) ‘you are here’  
  - notify other authors of adding the C to CM | • Chk (C, CM, ∃ ) = true  
• perform keyword search on C within DO (domain ontology)  
• U (CM, C) | • L (synonyms, C, CM)  
• L (DirRelC, C, CM)  
• L (NonDirLinks, C, CM) |
| Add (L, C1, C2, CM) | • suggest options for the author:  
  - add or delete domain engine results  
  - notify if one of the concepts is missing from the CM  
  - notify if the link exists and offer editing option  
  - notify about an indirect linking path between the C1 and C2 and give altern. Present. Of DE results:  
    - V (Text, ReC, Relevance %)  
    - V (Graph, Domain Ontology, Highlighted paths)  
  - notify other authors of adding the L to CM | • Chk (C1, CM, ∃ ) = true,  
• Chk (C2, CM, ∃ ) = true  
• Chk (L, CM, ∃ ) = true  
• Chk (C1-x-C2, CM, ∃ ) = true (indirectly linked with path ‘x’)  
• U (CM, L) | • L (x, C1-C2, CM) |
| Del (L, C1, C2, CM) | • give alternative presentation of all NonDirLinks:  
  - V (Text, L, Relevance %)  
  - V (Graph, Domain Ontology, Highlighted paths)  
  - Give user the option to either graphically or textually cut all segments or sub-sets of them  
  - notify other authors of removing L from CM | • Chk (L, CM, ∃) for L in CM = true  
• Del (L) and U (CM) | • L (x, C1-C2, CM) |
| Del (C, CM) | • Notify the author if C has DirLinks  
• Give alternative views with the option to delete them or cancel the delete action:  
  - V (Text, DirLinks, Relevance %)  
  - V (Graph, Domain Ontology, Highlighted paths)  
• notify other authors of removing C from CM | • Chk (L, ∃ for C in CM) = true  
• Del (C, CM) and U (CM) | • L (DirRelC, C, CM)  
• L (DirLinks, C, CM) |

A concept combines pages into larger conceptual units. This way the notion of prerequisites is defined (in regard to user’s knowledge acquisition based on reading pages). The structure of related fragments, pages, and abstract concepts forms a three-level hierarchy: (1) fragments are at the lowest level, (2) pages are at the middle level, and (3) abstract concepts are at the highest level. Pages are the units of information presented to the user on each iteration. Pages are bundled into abstract concepts, which can build higher-level abstract composite concepts. The concept relationships represent semantic (non-navigational) relationships between concepts [6].

5.2 Basic Domain Authoring Tasks in AHA!

The basic DM authoring tasks defined in AHA! include:

- Add a concept to the list of concepts.
- Delete a concept from the list of concepts.
- Define a relationship.
- Link concepts with a selected relationship.
- Choose a view on the graph (filter).
- Save the current concept collection together with the relationships in an XML file.
- Load already existing XML file with concepts and relationships.
- Export the concept collection in an appropriate format to AHA!.

As a specific example let us consider the task ‘add concept’. A concept in AHA! is represented as follows:

```xml
<concept>
  <name> brendan </name>
  <description> </description>
  <resource> </resource>
  <requirement> </requirement>
  <attribute name="access" type="bool" isPersistent="false" isSystem="false" isChangeable="false">
    <description> </description>
    <default> </default>
    <generateListItem isPropagating="false"/>
  </attribute>
  <generateListItem isPropagating="false"/>
  <trueActions>
    <action>
      <conceptName>ad</conceptName>
      <attributeName>defined</attributeName>
      <expression>ad.defines + (1.0 * brendan.knowledge) </expression>
    </action>
  </trueActions>
  <falseActions>
    <action>
      <conceptName>ad</conceptName>
      <attributeName>defined</attributeName>
      <expression>ad.defines + (1.0 * brendan.knowledge) </expression>
    </action>
  </falseActions>
</concept>
```

It is clear from this definition that the authoring task ‘add concept to the list of concepts’ requires from the author actions very similar to those listed in 4.2. Consecutively, they can be represented by the same authoring subtasks:

- specify (fill-in) concept’s attribute
- indicate concept’s relation to another concept, which in turn includes:
  - select a concept
  - specify link’s attributes.
6. An Example: WATO DM Authoring Tasks Collection

A comparison between the domain authoring tasks involved in AIMS and AHA! as well as in some other WIS shows their similarity, which is in support of the suggested here framework.

Considering the atomic tasks within the base layer of WATO a hierarchy of basic DM authoring tasks can be defined. The following is an example of the hierarchy for the composite task ‘create domain’.

![Domain tasks hierarchy (base layer)](image)

Create domain is a composite authoring task, which consists of five basic tasks: 'add concept', 'add link between two concepts', 'define new concept attribute' and 'define new link attribute'. Each of them includes several levels of composite and atomic tasks, such as 'specify attributes', 'fill-in value', 'add object' or 'select existing object'. For reasons of space limit, Figure 5 exemplifies only the sub-tasks for the basic tasks ‘add concept’ and ‘add link to another concept’. Note, that in this example only the ‘sub-task-of’ relationships are presented. There is a more significant difference between the task model authoring in both systems and our efforts will be further focused on the development of this part of WATO.

7. Conclusion

Authoring of concept-based WIS aimed at providing adaptation or various sorts of personalization to the user is becoming more and more complicated than the ‘standard’ WIS authoring thus requiring more intelligent support. An approach to such support with regard to both application authors and system developers should provide a link between high-level author’s tasks and low-level developer’s functions. In this paper we propose an ontological (in compliance with the Semantic Web technologies) approach towards a common authoring framework to formally describe the process of presentation generation in concept-based WIS. By having knowledge of concept-based WIS authoring process and being able to decompose it into steps and procedures and reason over it, the authoring support system itself will be able to perform various (semi-) automatic actions and to provide hints and recommendations to the author. A key idea of the proposed approach is to separate completely and clearly the content and the application related issues within WIS, which will allow for better modeling of the application-related processes and for making accessible and retrievable external data in adaptive (user-oriented) WIS.

References

1. Introduction

In the AI in education research community and related academic communities such as learning science (LS), instructional science (IS) and instructional design science (ID), research results have already accumulated and many sophisticated learning support systems, interactive learning environments, and intelligent tutoring systems have been built to date. One problem, however, is that it is not easy to build a good learning support system well-justified by such basic theories because they are not easily accessible for system designers/developers nor ready for engineering use. Imagine a theory-aware authoring environment, which can help designers/developers find, understand and utilize necessary LS/IS/ID theories[1,2,3] to build a theory-justified learning support system. It would give a considerable impact on this community.

On one hand, we need to articulate and organize the knowledge exposed in LS/IS/ID theories and to implement it in a computer for engineering use. On the other hand, knowledge of the architecture and building process of various types of learning support systems also must be declaratively modeled and implemented. And then, the correspondence and relationship between both of the knowledge is explicitly represented declaratively. This is what such a theory-aware authoring environment has to have. In AI research, ontological engineering, which tells us how to extracts and organize concepts in a target world declaratively in a computer-understandable form is becoming available. This is why ontological engineering approach could play an important role in our research.

This paper is organized as follows. The next section overviews what have done to date by members of our research teams. Section 3 presents a summary of each development such as an ITS ontology, Training task ontology, OGF (Opportunistic Group Formation) ontology, an ontology of function for systematization of functional knowledge used in mechanical design and LS/IS/ID theory ontology followed by concluding remarks.

2. A brief overview of ontology development

Our research team has some experience in building ontologies for learning support systems and LS/IS/ID knowledge as well as a road map towards a theory-aware authoring system. On the basis of the development of a generic framework of an ITS, a comprehensive ontology for ITS was designed [4]. The ontology has developed into the following SmartTrainer task ontology [5], which contributed to realization of an ontology-aware authoring tool for a training system.

The next ontology that was built is one for Opportunistic Group Formation (OGF), which dynamically forms a group for collaborative learning based on learning theories. The OGF ontology [6,7] succeeded in making some Learning theories available to engineers.

The next ontology is concerned with functional knowledge in mechanical design. Although the domain is not education-related, the methodology used in the research is expected to make a reasonable contribution to the LS/IS/ID theory ontology building because both are the same in that they are efforts for knowledge systematization through ontological engineering [8]. The last one the authors currently design is a bit more ambitious and comprehensive in that the ontologies will cover learning, instruction and instructional design theories [9,10].

3. Some examples of ontologies

3.1 An ITS ontology [4]

An ITS ontology was built based on the following observations: Building an ITS (Intelligent Tutoring System) requires a considerable amount of work. However, it is most often built from scratch. Little functional components are reusable and we rarely compare or assess the existing systems. In order to design an ITS, the designer has to know what an intelligent educational system is, that is, what functions are necessary for his/her goal of education, what components are necessary for what functionality, how to specify each component, what architecture is appropriate for the goal, how to control the behavior of the components, etc. Issues here include how much we know about these fundamental characteristics of an ITS. Although there have been much philosophical discussion and implementation of ITSs, there is little in between the two. What we need for filling up this gap are well-designed common vocabulary and a framework for educational systems. We also need to
formalize educational tasks at the right level of abstraction. We need to explicitly define learner's and system's roles and tasks in educational processes. An ontological engineering seems to be a good solution to this problem.

An ontology consists of task ontology which characterizes the computational architecture of knowledge-based systems and domain ontology which characterizes the domain knowledge. Task ontology provides us with an effective methodology and vocabulary for both analyzing and synthesizing knowledge-based systems which an ITS belongs to.

The top-level categories of task ontology of ITSs consist of Goals of education, Learner's state, System's functionality, Learner-system interaction, and Teaching material knowledge because an ITS is characterized as an interaction between a system and a learner in which the system's activity is based on its functionality which is performed in a domain according to the learner's state under a certain goal. Then, we have the following top level categories of concepts.

- Goals of education
- Learner's state
- System's functionality
- Learner-system interaction
- Teaching material knowledge

Details of the ontology is found in [1].

3.2 Training task ontology[5]

The above mentioned ITS ontology has developed into a training task ontology which has been used for building an intelligent authoring tool (See Fig. 1). We consider creation of a training material for machine operation such as power plant operation requires the following three procedures:

- Building training scenarios
- Developing a simulator of the target machine
- Deciding the learning items appearing the scenarios

Due to the procedures listed above, the authoring process of our ontology-based intelligent training system should:

1. make the concepts for creating training materials and the interface clear. (Done by the ontology author with an ontology editor)
2. define the axioms of concepts. (Done by the ontology author)
3. make the models of training scenarios (Done by the end author) at the conceptual level based on the training material ontology, and then compile those models so that the authoring system can check the coordination of the models according to the axioms defined by the ontology author.
4. Execute those models at the conceptual level to see the result.
5. After the repetition of (3)&(4), develop the final training materials in details by creating simulation cards and learning item cards based on the conceptual level models, and compile the final training materials to check if there is an undefined necessary card.

According to the authoring process shown above, the construction of our intelligent authoring tool consists of authoring interface, model compiler, conceptual level execution module, teaching materials compiler, and ontology construction environment. In these components, ontology construction environment contributes to (1)&(2), model compiler contributes to (3), conceptual level execution module contributes to (4), and training materials compiler contributes to (5).
3.3 OGF ontology[6,7]

The next ontology to be built is based upon the idea of “Opportunistic Group Formation” and can be expressed as follows:

Opportunistic Group Formation is a function to form a collaborative learning group dynamically. When it detects the situation for a learner to shift from individual learning mode to collaborative learning mode, it forms a learning group each of whose members is assigned a reasonable learning goal and a social role which are consistent with the goal for the whole group.

We briefly explain the outline of the OGF. Figure 2 shows the overview of the OGF ontology. Negotiation ontology is a kind of task ontology and is one for negotiation task. It is independent of the subject of negotiation. Collaborative learning ontology is an ontology for collaborative learning and is also independent of subject domain. It is composed of several ontologies one of which is Learning goal ontology. Learning goal ontology is designed by analyzing learning theories for group formation in a collaborative learning setting. Figure 3, in which I-goal, Y<=I-goal and W-goal stand for the goal of a particular learner, the goal of interaction of a particular learner with another learner and the goal of the whole group, respectively and G:Y(Lb)<=I(La) for the goal of learner La’s interaction with learner Lb, shows the key concepts in the ontology which were extracted from typical learning theories such as peer tutoring, anchored instruction, LPP, etc. One of the common factors across theories is that each of them has its own role assignment to the participants with specific goal associated with the roles. This turns out Learning goal and associated learning goals specifies collaborative learning

3.4 Knowledge systematization using an ontology[8]

The topic of the following ontology is on systematization functional knowledge in computer-aided design[8]. The domain is different from educational knowledge. The reason why we discuss the issue here is to demonstrate the effectiveness of ontological engineering by showing how functional knowledge systematization through ontological engineering has been successful, which is expected to be applicable to LS/IS/ID knowledge systematization.

Knowledge systematization is indeed a topic of content-oriented research and is not that of a knowledge representation such as production rule, frame or semantic network. Although knowledge representation tells us how to represent knowledge, it is not enough for our purpose, since what is necessary is something we need before the stage of knowledge representation, that is, knowledge organized in an appropriate structure with appropriate vocabulary. This is what the next generation knowledge base building needs because it should be principled in the sense that it is based on well-structured concepts with an

![Figure 2 OGF ontology.](image)

![Figure 3. Goals of group learning.](image)
explicit conceptualization of the assumptions. This nicely suggests ontological engineering is promising for the purpose of our enterprise.

While any scientific activity which has been done to date is, of course, a kind of knowledge systematization, it has been mainly done for human consumption. Our knowledge systematization adopts another way, that is, ontological engineering to enable people to build knowledge bases on the computer as a result of knowledge systematization. The philosophy behind our enterprise is that ontological engineering provides us with the basis on which we can build knowledge for computer consumption. By building a framework for knowledge systematization using ontological engineering, we mean identifying a set of backbone concepts with machine understandable description in terms of which we can describe and organize design knowledge for use across multiple domains. The system of concepts is organized as layered ontologies as is seen in Figure 4.

**Functional modelling.** Any artifact can be as it is thanks to its functional integrity. The concept of function therefore plays a crucial role in the mechanical design world. One of the key claims of our knowledge systematization is that the concept of function should be defined independently of an object that can possess it and of its realization method. If functions are defined depending on objects and their realization, few functions are reused in different domains. As is well understood, innovative design can be facilitated by flexible application of knowledge across domains.

Although functional representation has been extensively investigated to date, most of the representation schemes are ad hoc and lack generality and consistency, which prevents knowledge from being shared. This is because it is not well understood how to organize functional knowledge in what principle in terms of what concepts. One of the major causes of the lack of consistency is the difference between the ways of how to capture the target world, that is, how to view a function. The difference will be one of the causes of inconsistency in functional representation and non-interoperability of the knowledge when functional knowledge from different domains is put into a knowledge base. These observations show that we need a framework which provides us with a viewpoint to guide the modeling process of artifacts as well as primitive concepts in terms of which functional knowledge is described in order to come up with consistent and sharable knowledge.

**Hierarchy of functional knowledge and ontology.** Figure 4 shows a hierarchy of functional knowledge built on top of fundamental ontologies. The lower layer knowledge is in, the more basic. Basically, knowledge in a certain layer is described in terms of the concepts in the lower layer. Top-level ontology defines and provides very basic concepts such as time, state, process and so on. Causal ontology specifies actions and causality against teleology. Physical world ontology specifies 3D space and entity to give axiomatic physical world with a state-based modeling reflecting a special world of design in which an entity(artifact) is created from nothing. These two ontologies contribute to “Symbol grounding” of higher-level concepts, that is, functional concepts. On top of these three, process ontology is introduced to specify natural processes or phenomena. Every device utilizes several natural phenomena to realize its functions.

The extended device ontology views an artifact as something that inputs, process and outputs objects. The object is something processed by the device during it goes through a device and hence it never be another device that cannot go through a device. This ontology imposes a proper viewpoint from which one can successfully model a mechanical system in a way consistent with those models of engineering artifacts produced in other domains. It is not an easy task to build models of a lot of artifacts in a consistent way. “A gear pair changes torque”, “A cam shrinks a spring” and “A cam pushes up a rod” are inconsistent with each other in the hidden computational models. While the first one is based on the extended device ontology, the latter two are based on a different ontology, say, inter-device operation ontology. The organization of knowledge including these models will lose consistency. The extended device ontology thus allows us to build interoperable models and provides us with a guideline for modelling process by its role-assignment functionality, the very source of consistency in functional knowledge organization.

Functional concept ontology specifies functional concepts as an instance of function defined in device ontology. The definitions are scarcely depends on the
device, the domain or the way of its implementation so that they are very general and usable in a wide range of areas. Theories and principles of physics and abstract part library also belong to this class of knowledge called general concept layer.

Functional achievement way knowledge is such knowledge that represents various ways of achievement of a function. This knowledge is about how (in what way) a function is achieved, whereas the functional concept is about what the function is going to achieve. In other words, the former is formulated in terms of whole-part relation and the latter in terms of is-a relation. Although functional achievement way knowledge looks similar to the conventional functional decomposition, the former is much richer than the latter in that it consists of four kinds of hierarchies of different roles and principles (is-a hierarchy, attribute tree, functional decomposition tree and general functional decomposition tree). The inherent structure of such knowledge is organized in an is-a hierarchy from which the other three structures are derived according to the requirement. The is-a structure is carefully designed identifying inherent property of each way to make it sharable and applicable across domains. One of the key issues in knowledge organization is clear and consistent differentiation of is-a relation from other relations such as part-of, is-achieved-by, etc. keeping what is the inherent property of the target thing in mind.

**Use of functional concept ontology.** On the basis of the above ontology, a new framework for representing functional knowledge for a machine has been developed and deployed into a production line at the Production Systems Division of Sumitomo Electric Industry, Ltd. It has been successful and typical results are shown below.

1. The same document can be used for (a) redesign, (b) design review, (c) patent writing and (d) troubleshooting
2. Patent writing process is reduced into 1/3
3. Design review goes extremely well
4. Troubleshooting is done much easier than before
5. It enables collaborative work among several kinds of engineers such as a researcher, a designer, a manufacturing engineer, an operator, a maintenance engineer, etc.

As a result, Sumitomo people have decided to commit to the ontology and to deploy the framework company-wide. They also would like to see a possibility to set up the standard functional knowledge representation for production engineers.

**Lessons learned.** What we have learned is summarized as follows:

(a) To investigate things from the basic concepts is essentially important

(b) One should find a set of key concepts and their relations in the task/domain

(c) These concepts contribute to identify the hidden but common concepts and conceptual structure of the target domain/task

(d) It is possible to systematize knowledge across domains

In summary, ontological engineering is useful to knowledge systematization to make the knowledge sharable and reusable across domains.

### 3.4 LS/IS/ID ontology [9,10]

Let us come back to the educational domain. An analysis of the existing authoring environments in terms of components shows that few of them combine authoring tools and knowledge representation of instructional theories and principles, and that none of them possesses desired functionalities of an intelligent authoring system such as Retrieve appropriate theories for selecting instructional methods or Provide principles for structuring a learning environment.

Declarative knowledge is mainly absent in those systems, as is the maintenance of its integrity. Ontological Engineering has the potential to solve these problems by proposing a declarative knowledge modeling approach; as a result, the semantic-based knowledge systematization could provide a gateway to learning objects and their management. Instructional Design theories provide the principled knowledge to make higher level design decisions such as instructional strategies, or to orient the lower level decisions such as learning material. However, the
systematization of instructional knowledge in the form of declarative knowledge is a challenge in itself.

The enterprise on LS/IS/ID knowledge systematization is totally based on the successful experience stated in the above subsection. A series of ontologies are being designed based on the nesting structure of the three theories, which gives effective access to declarative knowledge (Fig. 5). The authors are going to build ontologies not for theorists but for designers or engineers who need effective help based on the theories when they build a learning support system or an interactive learning environment. This requires the ontologies be operational to organize these theories from engineering point of views. This is the key success factor of our enterprise, which is otherwise almost impossible to achieve, since each theorist has his/her own specific views and terminologies are more or less compatible to others.

The engineering point of view enables us to apply an instructional design methodology that can be oriented by theories, which is used for specifying the needs for it and contributes to establishing the common conceptual background on which the ontologies can be designed. Another merit of the viewpoint is that it allows us to approximate each theory to put it into the engineering system framework. This is also one of the success factors of the enterprise, since theories are sometimes ignored by designers/engineers. Viewing the authoring task as part of instructional design, where design decisions are made, access to theoretical knowledge should improve quality and consistency of the design. Our effort to systematize theoretical knowledge and to integrate it into a design methodology for authoring tasks is, as far as we know, an original contribution to the field of AIED.

Theory-aware authoring system. One of the difficulties the current situation has is theorists tend to insist that the older theories are bad or wrong, theories are incompatible to each other because of the different conceptualization of the target worlds (LS/IS/ID), etc. Unfortunately, however, such statements are not correct from engineering and ontological points of view. Viewing from the both points of view, theories are compatible each other and each of which has its own goal and own application situations which are different from those of the others. The important thing is that from the ontological point of view, it is possible to come up with the common concepts and a conceptual structure to explain existing theories in a harmonized way and from engineering point of view, we know we need all the goals and situations the existing theories cover depending on the user requirements.

We view the three worlds, LS, IS and ID worlds, according to the nested structure shown in Fig. 5. Learning theories explain what type of learning occur under what conditions, instruction theories explain how to facilitate such learning in terms of instructional processes and events, and instructional design theories explain how to design instructional processes/events. This view helps us “ontologize” the three worlds.

4. Concluding remarks

We have reviewed what we have done thus far followed by on-going project on "LS/IS/ID knowledge systematization from engineering point of views through ontological engineering". Further steps on our roadmap include:
1) Produce one or two detailed examples. We currently develop a minimum ontology for demonstrating how our system works to help an author build a lesson scenario using three different scenarios built by experts.
2) Elaborate functionalities of an ontology-based ITS authoring system. A long term perspective for this work is to provide the next generation of authoring systems with a scientific basis for semantic standards of learning objects and their management.

Reference
Automatic Construction of Learning Ontologies

Trent Apted, Judy Kay
School of Information Technologies, University of Sydney, Australia, 2006.
{tapted, judy}@it.usyd.edu.au

Abstract

This paper describes a system that automatically constructs an extensive ontology of computer science. This can serve as a basis for making inferences about student models and other reasoning within a teaching system. The system enables a user to select a focus concept and then see the most closely related concepts. This paper will also describe possible machine learning applications of the ontology, for providing a conceptually concise basis for developing and communicating knowledge, and the tools constructed to facilitate this.

1. Introduction

A software representation of an extensive computer science ontology may be used in a variety of roles in a teaching system. It enables a system to infer from a minimal knowledge base about a student to a quite extensive model of their knowledge. Teaching goals can then be inferred by studying this model and expanding concepts the student is modelled as knowing – working with those that are closely related. It is also possible for a learning goal to be established, with the ontology identifying aspects that need to be taught as a foundation for this goal; for example, if they are prerequisites.

The size and scope of such an ontology means that it becomes necessary to effectively query the information it represents. This also allows a model within the scope of the ontology to be constructed efficiently using minimal knowledge. Models can be compared, looking for similarities, and a result given that reflects their relative structure and position in the ontology. This result can then be used as additional information to learn from the models.

Rather than creating the ontology by hand, we have built a tool for automatically constructing an ontology, dubbed MECUREO1. It uses an existing, reliable resource to ensure appropriate coverage of computer science terms. This saves a considerable amount of effort and helps minimise errors while maximising breadth of coverage. The generated ontology may also serve as a basis for refinement by hand. For example, it is often easier to categorise an unknown relationship (that is, a relationship known to exist for which there is not enough information to categorise it) by examining the automatic construction than it is to identify a relationship independently.

Our approach to the ontology construction overcomes some of the limitations of existing ontology tools and representation formats such as OilEd [1] and DAML [2]. Specifically, an incorporation of weighted relationships between concepts allows more information to be extracted from the ontology towards learning goals. In addition, the representation format is customised to improve the handling of such a large ontology efficiently and effectively.

2. Ontology Construction

In order to construct the ontology, the resource chosen was the Free On-Line Dictionary of Computing (FOLDOC) [4]. This is a free Internet resource that currently contains definitions for 13,590 computer science terms in over 4.8 megabytes of text; it is quite extensive. The off-line version encloses significant words (eg words defined elsewhere) in braces and uses standard conventions in the layout and grammar of each definition; an example of which is given in Figure 1. This makes it feasible to perform automatic analysis of the definitions.

FOLDOC was chosen because of its breadth of up-to-date coverage within the computer science domain, as well as its machine readability and licensing arrangement. In the entry in the dictionary under Free On-Line Dictionary of Computing, the dictionary describes itself as “a searchable dictionary of acronyms, jargon, programming languages, tools, architecture, operating systems, networking, theory, conventions, standards, mathematics, telecoms, electronics, institutions, companies, projects, products, history, in fact anything to do with computing”. Its focus is providing a one-stop source of information about all computing terms. It includes many cross- and bibliographical references and is kept up-to-date with emerging technologies and institutions.

The ontology itself is generated by processing an off-line image of the FOLDOC computer science dictionary. Individual definitions are parsed, looking for a categorisation for each word and to identify words that are related to it. Grammatical conventions in the dictionary are also used to provide additional information to the relationships wherever possible.

The consistent grammatical conventions mean that synonym, antonym, child and parent relationships can be extracted if certain keywords are present; otherwise an

---

1 Originally, "Model Expansion and Comparison Using Reverse Engineered Ontologies"
unknown relationship is created. For example, acronyms usually become modelled as synonyms of their expansions, relationships identified such as between call-by-value and call-by-name are modelled as antonyms, patterns such as developed at X or a type of X represent child relationships, and a list of cross-references after an entry is itself defined; by default these also become nodes or can optionally be ignored. In our example, a link in an entry will be considered in a query that visits the node ontology – random in the sense that the decision to use it is not based on how MECUREO performs with it – and one not used to calibrate the system. In the discussion that follows concept is used to describe an entity at the ontology (conceptual) level, word at the dictionary level and node at the representation level; similarly for relationship, link and edge with respect to entity associations.

2.1 Determining the Nodes

Consider the entry in the dictionary for ontology shown in Figure 2; parsing this entry initially results in the node ontology being generated. It becomes linked with the category nodes for philosophy, artificial intelligence and information science, such that a query on any of these categories will result in the ontology node being returned. All phrases in braces (\{AI\}, \{knowledge\}, \{domain\}, etc.) also become nodes and an edge is formed between each and ontology.

It should also be noted that, despite an inherent direction (with the node whose definition's body the link was extracted from designated the head), the edges are traversed in either direction for the purpose of querying. This means that the node for any dictionary entry that referred to ontology will be considered in a query that visits the ontology node. In addition, it is sometimes the case that a link in an entry is not itself defined; by default these also become nodes or can optionally be ignored. In our example, subject indices and Knowledge-Level are two such links.

The final output of the parsing process is backed by a weighted digraph. Further properties are recorded with the nodes and edges to facilitate querying. The procedure to obtain this output, given FOLDOC and the keyword-relationship mapping as input, will be discussed in the context of the entry for ontology in Figure 2. A visual representation of the output corresponding to this context is shown in Figure 3. This context was chosen to try and demonstrate a typical entry – random in the sense that the decision to use it is not based on how MECUREO performs with it – and one not used to calibrate the system. In the discussion that follows concept is used to describe an entity at the ontology (conceptual) level, word at the dictionary level and node at the representation level; similarly for relationship, link and edge with respect to entity associations.

2.2 Determining the Relationship

For successful querying of such a large ontology, it was found that, in general, it was unproductive to treat all
relationships between concepts in the same way. It was decided that each relationship should be given a weight that reflects the amount of work required to 'travel' from one concept to the next. The first step towards this is determining that a relationship exists and its type, after which a weight can be inferred.

The type for each edge is determined from keywords around the phrase and the currently loaded relationship mapping. When a link can be designated a type in this way, it becomes a property of the edge so that it can also be used in subsequent processing and output (eg visually). The keyword also determines an associated direction of the edge, based on the type.

It is often possible to identify a link as being an antonym, or strictly a child or parent, and this information is preserved in the graph. For our experiments, these mappings were determined by manually analysing the grammar used in a selection of definitions in the dictionary. Any patterns recognised were recorded so they could then be used to classify the links in all the definitions of the dictionary (see §0).

For example, keywords identified in the extract in Figure 2 include formal, see, e.g. and about. See gives a type to the edge with subject index and about with (all three of) domain, declarative language and universe of discourse. Here axioms is an unfortunate anomalus classification; e.g. and formal both appear near axioms and because e.g. was configured to have a stronger base weight it is used for this edge rather than formal, giving a 'child' edge rather than the 'parent' type associated with formal. This is all partly reflected in Figure 3: subject index has a bidirectional (sibling) edge and axioms has a directed 'child' edge. About was not configured with a strong enough base weight to have a distinctive edge in this output configuration.

2.3 Determining the Weight

The base weight for the determined link type is a constant between 0 and 1 specified in the configuration. After assigning this weight to the edge, it is then adjusted (penalised) depending on the position of the phrase in the definition. If it occurs later in the definition, as determined by the number of links identified since the last category phrase, it is given more weight.

The heuristic used for the adjustment $p_i$ of the weight for the $(i+1)^{th}$ link in the definition is

$$P_0 = 0, P_{i+1} = p_i + \frac{1 - p_i}{C}$$

Here $C$ is an adjustable positive constant, greater than one, to control the rate of change (10 is the default). A small $C$ will cause later links in highly branching nodes to be penalised more, while a large $C$ will distribute the weights for successive links more gradually. This form of the heuristic shows more clearly how the increase for each $i$ approaches zero as $p_i$ approaches 0.5; rearranging with the default of 10 yields simply $p_{i+1} = 0.05 + 0.9p_i$. Otherwise, the general closed form is

$$\frac{p_{i+1}}{2} = 1 - \frac{p_i}{C}$$

Essentially it gives a gradually increasing penalty smoothly tapered at 0.5.

This formula is a compromise in order to assign 'fair' weights to all links (both in definitions with many links and those with few) and was found to facilitate lucrative querying. Generally, a problem was found when using more discrete weights when it came to performing queries on highly branching nodes. Because the number of nodes within a certain distance of a node is proportional to the exponent of the distance, the size of a query can increase rapidly. Discrete weights, or a truncated penalty function, meant that the number of nodes returned in a query could not be finely tuned.

The formula was used to spread the distribution of weights using the information that was available. That is,

![Figure 3. Point query from ontology with a depth limit of 0.9](image)

*Bidirectional edges indicate synonym (or strong sibling) relationships, reversed arrowheads indicate antonym (or other opposing) relationships, directed edges indicate strict parent/child relationships and undirected edges indicate undetermined relationships (or weak siblings). Bold, normal, dashed and dotted line styles indicate progressively weaker relationships for all types. These are discussed further in §0.*
links mentioned early in a definition are generally more closely related to the word being defined. This is a rather bold generalisation, but it is reflected directly in many definitions in the dictionary. For example, from the ontology extract in Figure 2, AI, knowledge and domain occur before universe of discourse, axioms, agents and Knowledge-Level and so the edges to the former are given less weight. In these cases it is intuitively the right thing to do and while there are many exceptions (eg logical theory in Figure 2), the generalisation was found to work well in our initial precision evaluations.

The final weight assigned to an edge is the sum of the base weight and the adjustment. It is possible for the weights of links whose type is explicitly weak to exceed 1.0, so this value is capped at 1.0. This allows a query to deliberately visit (at least) every node at a particular depth (as opposed to distance) without additional computation being required.

There was initial concern that this might be counter-productive. Having a large number of edges with a weight of 1.0 ignores the arguments for having a smooth penalty function (above), but it was found not to be the case in practice. Explicitly weak relationships are a minority as it is rare for a keyword (when one is actually found) to indicate a conceptually weaker relationship. The words in the context of truly weak relationships are usually too obscure to improve the overall performance, as we discuss below. However, the penalty adjustment was often found to weight these relationships appropriately.

On top of this, there are some special cases. In the extract for ontology in Figure 2, AI, for example, was not adjusted because it is the first relationship identified (\( p_i = 0 \)). So this edge retains the default base weight, but if we then look to the definition of AI in Figure 4 then another pattern is identified – a synonym. So an edge with negligible weight is formed between AI and artificial intelligence to reflect this property. In this way anything that is close to AI is also close to artificial intelligence and vice versa.

![Figure 4. The definition for AI](image)

Links occurring in the body of a definition whose type is determined to be very strong are similarly modelled (i.e. their base weight is zero). However, these are adjusted depending on their position in the same fashion as normal links.

The final result of this example is shown in Figure 3, as a point query, and is discussed in §0.

3. Ontology Queries

The ontology generated is very large – in excess of 23,000 nodes. Tools have been written to query the ontology from a specific concept, or a set of concepts (for example in a user model). This results in a subgraph of the original ontology graph being generated. As a useful side-effect, nodes (as well as edges) are assigned a weight that reflects how closely each node matches the original query. The multi-point queries executed from a set of concepts (for generating models) and merge queries (for combining and comparing models) are still in evaluation and (re)development stages and will be discussed in a later paper.

3.1 Point Queries

A query can be executed from a single node (which may be word, phrase or category) in the original graph, or in any graph generated as the result of a previous query. This query results in a concept map style representation of all things related to the query node within a certain distance. This is useful for browsing the ontology and for general information retrieval, and was the primary method used to evaluate the ontology construction.

The distance is specified as part of the query and its reach is determined by the edge weights corresponding to the relationships between nodes. All nodes whose shortest path to the query node is less than or equal to the specified distance are included in the output graph. This is determined in increasing order of path cost, so it is also possible to specify a node limit to restrict the size of the graph. The shortest path cost becomes a property of each node, which can then be used towards a visualisation heuristic.

Once the nodes are determined, any edges that exist between them (not only those along the shortest paths) are duplicated. This preserves the structure of the ontology to facilitate further querying on the output graph, to perform structure analysis (eg for comparisons), and in order to represent all the information available when the graph is displayed. An example of a point query is shown in Figure 3 from the node ontology. This also shows the result of parsing the dictionary extract given in §0. Bold, bidirectional edges indicate relationships identified as being synonyms (or other immediate cross references in the dictionary). For example the relationships between NN and neural network, and between neural network and artificial neural network, are identified as synonyms. Bold edges with reversed arrows (eg between declarative language and imperative language) represent antonym relationships. Other bold edges indicate very strong relationships, as determined by the keyword-relationship mapping. Directed edges attempt to identify strict parent-child relationships. However, they are also used for strong links where an explicit type could not be determined (the definition that contained the original link becomes the child). Edges with no arrows represent relationships for which a type could not be determined, with dashed and then dotted line styles\(^2\) representing progressively weaker relationships, as determined by the assigned edge weight.

\(^2\) See Figure 2 for an example of dotted line styles
Figure 5. Point query from ontology with a depth limit of 1.49 and a minimum peerage of 12

When there is more than one edge between two nodes, this indicates more strength, because the definitions for each of the nodes will have referred to each other.

In Figure 3, the depth limit of 0.9 is restricting the range of the nodes included. All the nodes related to ontology are included because there are no weak relationships with ontology and 1.0 is the weight limit for a single edge. However, for the nodes related to ontology, only synonyms or other strongly related peers are included, such that there is always a path from any node to ontology with a cost less than or equal to 0.9.

A deeper query, also from ontology, is shown in Figure 5. As you would expect, the number of nodes is exponentially proportional to the depth of a query due to branching. In order to restrict the output so that it remains relatively small and uncluttered, a minimum peerage of twelve is enforced for this figure. Highly connected nodes are generally less obscure, so are of more interest to someone viewing the model, which is why this filtering method is effective.

Viewing the ontology in this way allows the 'bigger picture' to be seen. From here, further detail may be sought, and this can be accomplished with a follow-up query on any node, using smaller values for the depth limit and minimum peerage. These queries can also show interesting chains of relationships, along the lines of the 'degrees of separation'. In fact, over 90% of the ontology is within a distance of 3.0 from some highly branching nodes such as SQL. This means that at least 90% of the nodes in the ontology are within a maximum distance of 6.0 from each other. See also §0 and Figure 6.

4. Evaluation of the Ontology

4.1 Tuning the Construction

In order to tune the construction procedure, a collection of randomly and deliberately chosen definitions were selected to intuitively evaluate the output and adjust the parameters. This began with selection of representative definitions, which were manually processed to determine the keywords and relationships to derive from them. The base weights and weight function were also adjusted to achieve an intuitively correct result.

Once the initial set of parameters were established, random definitions were queried and the results were compared with the definitions. Adjustments and additions were made to improve the correspondence between the logical structure of the definition and the resulting graph. The ontology was then rebuilt and changes were checked against all previously examined definitions. An error rate was established based on the number of incorrectly classified relationships.

A compromise had to be reached between the error rate compared with the definitions.
and the number of classified relationships. Each addition decreased the number of unknown relationships. However, when a change was made that resulted in the number of incorrectly classified relationships increasing more than the number correctly classified relationships, it was reversed. The test set was gradually increased in this way, with randomly selected definitions, until no more changes could be made that satisfied this requirement.

The final size of the test set, not including synonyms, was 193 definitions. This is roughly one percent of the entire dictionary. Further examination may improve the success slightly by adapting to patterns specific to small sets of definitions. However, this is essentially refinement by hand and should be performed on the ontology directly (rather than the automatic construction parameters) so that fewer errors are introduced.

Once the construction parameters were thus tuned, empirical evaluation was performed on the resulting ontology. These were conducted using only concepts not in the final test set.

4.2 Concept Mapping Experiments

The behaviour of the point queries lends itself nicely to comparison with manually constructed concept maps. A small set of volunteers (undergraduate computer science students) were given a number of topics (concepts in the ontology) closely related to subjects they had previously studied. They were then asked to draw a concept map around this concept, after being shown a mundane example of a concept map based around farm. There was no intervention with the volunteer until the exercise was complete.

The starting concepts were chosen to reflect the students’ area of expertise and encourage uninhibited expansion, as well as represent some key concepts in the ontology. The final choices were declarative language, SQL and device. This resulted in a collection of hand-drawn concept maps for each concept, which were compared against point queries in the ontology from the same starting concepts. The volunteers were then asked to comment qualitatively on the concept maps generated by the ontology.

4.3 Comparison with Trusted Sources

To perform a quantitative analysis, trusted knowledge bases were consulted. Most often these were significantly smaller, or had a different focus that was not appropriate for comparison with a computer science ontology. The main resource used to perform this analysis was the ASIS Thesaurus of Information Science [5]. This contains 1615 concepts from the information science domain arranged, hierarchically, in a tree. However, it touches upon many corporate concepts that the FOLDOC ontology does not cover (eg, one of the top-level nodes is business and management operations).

The flexibility of the construction tools was also put to the test in the evaluation method. Some simple pre-processing on the thesaurus was performed to convert it to a format similar to the dictionary. It was then parsed by the same tool used to construct the ontology. In this way all the existing querying and visualisation tools could be used to aid the evaluation procedure.

However, the structural elements of the thesaurus and dictionary are not the same. The thesaurus is a tree, and there is only one edge type, although the ordering of children in the thesaurus could be used to give weight to the edges. Evaluation was conducted by examining the distance in the ontology between a concept and its parents in the thesaurus. Stemming was used to assist matching of names.

5. Results

Presently, only partial results can be presented as work is still continuing. Also, it should be noted that all of the examples given in this paper, and those used to evaluate the ontology, were the direct result of the automatic construction. At no point were the structures of the graphs modified in any way from the generated result. Furthermore, the definitions corresponding to the nodes in these graphs were not used to tune the construction process, (eg in determining the keywords to use that decide relationship types). The obtained ontology’s intended use is as a foundation for further refinement by hand, in order to improve the performance of the ontology in specific applications.

Figure 7 - Concepts near ‘declarative language’ with a minimum of 10 peers
Using an image of the dictionary obtained on 2002-03-12, the following statistics were obtained:

<table>
<thead>
<tr>
<th>Definitions</th>
<th>Category links</th>
<th>Plus other links</th>
<th>Keywords processed</th>
<th>Preparse: 13536 defs for 153 categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>13536</td>
<td>8998</td>
<td>54830</td>
<td>33297</td>
<td></td>
</tr>
</tbody>
</table>

Graph Size: 23095, Density: 61926

The reason for the graph size being greater than the number of definitions is due to links being identified for words not actually defined, as well as for category nodes.

The performance of the queries with respect to the breadth achieved is indicated in Figure 3 and Figure 6. The exponential increase in query size with respect to distance is clearly seen in Figure 6. As the size of the ontology increases, the total number of nodes available, this relationship is clearly tapering, as expected, due to the limited scope of the ontology. Note that in this example, a particularly dense region of the ontology (SQL) was chosen as a starting point to achieve consistency at small distances. Most queries yield a distribution that starts slower and don't achieve exponential growth until a distance of approximately 0.7, showing the same decay at a greater distance.

Figure 5 and Figure 7 show the depth and far-reaching relationships that can be extracted from the ontology. These were constructed by filtering out the nodes that did not have a minimum number of peers when the graph definition was output.

The results of the concept mapping experiment are largely qualitative. Overall findings are summarised in Figure 8. Of the 122 total nodes drawn, 105 were exact or near matches to concepts in the ontology – around 86%. The remaining nodes were most often too specific instances of a concept, or were parts of the syntax used by a particular concept. Specifically, the following hand-drawn nodes were unable to be matched: screwdriver, spanner, mallet, flat head, Phillips head, rule, constraint satisfaction problem, iProlog, FROM, WHERE, SELECT, UPDATE, INSERT, 1:1, 1:M, M:1, M:M.

There was some discrepancy in the placement of edges. All but one participant's concept maps had a tree structure, despite encouragement beforehand to draw a graph. This contributed a little to the observation that often concepts drawn as being connected via an intervening node in the thesaurus does not correspond directly to that of the ontology.

Due to synonym and strong relationships, 90% of costs ranged between 0.2 and 1.9 units. The mean cost was approximately 1.4 units, translating to an average of around two edges on each path. The remaining 10% of costs were too large to have the nodes considered as being directly related due to branching effects, as described for above for Figure 6.

For each parent node in the thesaurus that matched a node in the ontology, a point query was performed. The cost assigned to each matched child (as a result of the query) was used to evaluate the precision. This cost is simply the shortest path, so this value was recorded for each node that matched. The analysis of these values was then conducted by hand.

Due to synonym and strong relationships, 90% of costs ranged between 0.2 and 1.9 units. The mean cost was approximately 1.4 units, translating to an average of around two edges on each path. The remaining 10% of costs were too large to have the nodes considered as being directly related due to branching effects, as described for above for Figure 6.

Figure 8. Concept mapping results

<table>
<thead>
<tr>
<th>Volunteer</th>
<th>Root Concept</th>
<th>Nodes drawn (excluding root)</th>
<th>Edges drawn</th>
<th>Nodes matched</th>
<th>Edges matched (depth ≤ 1.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A SQL</td>
<td>18</td>
<td>25</td>
<td>25</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>A device</td>
<td>28</td>
<td>28</td>
<td>23</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>B declarative language</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>C declarative language</td>
<td>19</td>
<td>22</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D declarative language</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E SQL</td>
<td>18</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>122</strong></td>
<td><strong>125</strong></td>
<td><strong>105</strong></td>
<td></td>
<td><strong>104</strong></td>
</tr>
</tbody>
</table>

6. Supplementary Visualisation

For small models (less than about one hundred concepts) a traditional graph output is able to provide useful
information to the user. To facilitate this, automatic output to the dot language [6] is provided. This output, along with the dot graph visualisation tool [7], was used to generate all the examples in this paper. It includes information regarding the relationship strengths and types, and positions nodes vertically according to how closely they match the query used to generate the initial graph.

7. Conclusion

This paper has described our approach to the automatic construction of a reasonably accurate and complete computer science ontology from a dictionary. The dictionary is parsed and a graph is generated with nodes representing concepts and weighted, directed edges representing the relationships in the ontology. The querying tools provide an effective way to focus on individual concepts or on a collection of concepts to represent a model, or models, to perform comparisons. Early evaluation has determined that the results are sensible, but further work needs to be conducted to determine the effectiveness of the ontology in its applications, such as for machine learning.

References
