Published in *Advances in Conceptual Modeling*. ER'99 Workshop on the World-Wide Web and Conceptual Modeling, Paris, France, Nov. 1999, LNCS 1727, Springer, 1999

Meta-Modeling for Web-Based Teachware Management

Christian Süß[1], Burkhard Freitag[1], and Peter Brössler[2]

[1] Fakultät für Mathematik und Informatik,
Universität Passau, D-94030 Passau, Germany,
{suess,freitag}@fmi.uni-passau.de,
WWW home page: http://daisy.fmi.uni-passau.de/
[2] software design & management,
D-81737 München, Germany,
peter.broessler@sdm.de,
WWW home page: http://www.sdm.de/

Abstract. In this paper we propose a meta-modeling approach to adaptive hypermedia-based electronic teachware that focusses on document structures and navigational services and which is also applicable to knowledge management.
An abstract meta-model is presented which is suitable to describe heterogeneous and semi-structured course material from different domains of application on the web. As an instance of this generic framework we derive a sample model for the domain of teaching computer science.
Content identification and querying at the meta-level and the use of metadata enhance navigation and facilitate adaptive presentation and navigation as well as reuse and adaptation of existing material to new audiences. Each model can serve as a well defined basis for a corresponding XML based learning material markup language (LM^2L) representation which can be restructured and rendered by XSL style sheets for different audiences, layouts, or platforms in web based teaching.

1 Introduction

Today, teachware (in this paper also called learning, teaching or course material) is frequently provided electronically on the Web in a variety of formats, e.g. as a collection of HTML pages, as PDF documents, as MS[1] Word documents, or as MS Powerpoint presentations, that can essentially only be accessed using the corresponding viewers or readers and thus is more or less unstructured. In most cases, meta information which could be exploited by a knowledge or teachware management system is either entirely missing or individually assigned in a rather ad hoc way and only at the coarse granularity of large units.

[1] Microsoft is a trademark of Microsoft Corporation

Conceptual and navigational modeling of teachware faces interesting challenges: On the one hand, learners may vary in their knowledge and interest and are seeking individual access to course material by adaptable navigation and presentation of given contents and structures [2]. On the other hand, it is essential for authors to find and combine existing learning material adapting it to new audiences or to share learning material.

Students taking a database course, for example, may want to get a first understanding of B-tree indexes without going very deep into the details. A typical question might therefore be: "Which definitions and formal results are needed for undergraduate students of computer science to understand B-trees from an application point of view?" Similarly, authors who are about to write a new course module about this topic and want to reuse existing material may ask for suitable figures and animations illustrating the formal material.

As the example above indicates, both types of users are looking for convenient navigation capabilities as well as for support of complex queries. In addition, authors should be supported by (re)structuring mechanisms, source (code) control, and version control. These similarities between software engineering and the development of learning material call for support of the life-cycle of learning material by a sound methodology, in particular with respect to the conceptual modeling.

In this paper we propose a meta-modeling approach to adaptive hypermedia-based electronic teaching material[2] which allows to describe knowledge about aspects of the contents of course material as well as navigational aspects.

In our approach each teaching domain has an underlying model that describes the navigational and the conceptual content structure used for this domain. We propose a sample model for the domain of teaching mathematics or computer science in which the latter could be given as a sequence of definitions, theorems, and examples. Other domains, such as teaching a foreign language, may require a different content structure which we will specify in corresponding models. At this point it should be noticed that in our approach the *document structure* is modeled rather than the conceptual structure of the subject or area the documents are about. This is far more than most of the existing electronic teaching material provides but still avoids the huge effort needed to conceptually model an entire topic itself.

To support navigation and retrieval as well as reuse and adaptation, the model defines domain-specific properties, i.e. groups of metadata, for (collections of) documents, conceptual units and conceptual relationships within. The conceptual relationships represented in the model are used as an ancillary structure [9] helping to navigate through the underlying material or to ask questions like "Which definitions are prerequisite to a given proposition?" thus giving the required support for adaptable navigation. Given a model which defines suitable

[2] It should be mentioned that, despite our focussing on electronic teaching throughout this paper, the techniques described are as well applicable to knowledge management in general.

conceptual units and their properties, it is possible to integrate material with a granularity ranging from single words to entire courses.

Furthermore, each model serves as a well defined basis for a corresponding XML based markup language, e.g. for the learning material markup language (LM^2L) [13] which we use to represent university courses in computer science. The metadata are used by XSL style sheets for restructuring and rendering the corresponding XML representation for different audiences, layouts, or platforms in web based teaching.

Our models serve as sophisticated a-posteriori data schemata that allow to apply database technology to web based learning and teaching to improve especially the access to documents.

As common basis for domain-specific models we present an abstract meta-model which is *independent of the underlying domain of application* and provides a *suitable platform to describe heterogeneous and semi-structured course material* from different domains of application on the web. The meta-model describes what it means to be a model, i.e. gives a definition of the general kind of structure description that is accepted and can be understood. This way we obtain an *extensible generic framework* which is *easy to modify and extend*: Teaching environments can be extended by new models for as different teaching areas as database theory or the world of opera. Furthermore, existing models can be easily extended to meet new requirements, e.g. by adding a *video clip* (see section 3.3). Both are important features for the rapidly evolving domain of web based teaching.

The rest of the paper is organized as follows: In section 2, we discuss common as well as domain-specific properties of teaching material and line up the requirements to teachware management systems. In section 3, a meta-modeling approach to teachware is proposed: We illustrate the different levels of our modeling architecture and present an abstract extensible generic meta-model. Using a sample instantiation, a model of the domain of teaching and learning computer science, we show the benefits of our approach to the management of teachware. In section 4 we introduce LM^2L, the Learning Material Markup Language and sketch the use of XSL to adapt learning material to different users. In section 5 we discuss related work. The paper is concluded with a summary in section 6.

2 Teaching Material: Properties and Requirements

2.1 Properties of Teaching Material

General Properties Semantically, teaching material consists of different conceptual units. There are course units but also training units as well as annotational units, etc., with the following properties:

Conceptual units exist at various levels of granularity and have an inner structure ranging from semantically unspecified floating text to semantical content objects. Conceptual units as well as the content objects within have relationships to other objects and units. These relationships are of different semantical types and are not bound to the same kind of conceptual unit.

Looking at teaching material, we have to consider navigational aspects concerning access to and navigation in the given conceptual units, as well:

Conceptual units can be presented to the user in various ways, e.g. as a (sequence of) HTML page(s) or pages in a PDF document. There are different types of access and navigation: Teaching material usually is accessed and navigated hierarchically via a root node or sequentially starting at a first node. However, there are also indexes, glossaries, and the like that allow direct, non-hierarchical access. Conceptual relationships usually can be presented as hyperlinks, providing an specific operational behaviour.

Finally, additional information (metadata) are assigned not only to (sequences, hierarchies, etc. of) entire conceptual units but also to (some of) the content objects and relationships they contain.

Domain-Specific Properties Whereas the properties mentioned above seem to be common to teaching material of different domains of application, there are domain-specific kinds of conceptual units, content objects and relationships as well as navigational units and access structures. Domain-specific types of hyperlinks show different kinds of operational behaviour and there exist domain-specific kinds of metadata.

2.2 Requirements to Teachware Management Systems

Teachware management systems[3] address two different groups of users: authors and learners. Both of them need support for thematic or typespecific filtering to provide individual access. They are also looking for adaptable navigation and presentation of given contents and structures, not only at the coarse granularity of entire courses or chapters, but rather at the fine granularity of content objects. Furthermore, convenient navigation using ancillary structures and sophisticated querying capabilities are important, as well.

In addition, authoring tools are necessary for creating new material and integrating existing one which often is heterogenous and only semi-structured. Finding 'similar' material, reuse, (re-)combination, adaptation, and sharing of existing material as well as controlling sources and versions should be supported, too.
In general, a teachware management system should be able to manage teaching material from different domains of application using database technology which turns out to improve especially the access to huge amounts of documents [1].

3 A Meta-Modeling Approach to Teaching Material

3.1 Modeling at Different Levels

The properties and requirements described in the previous section emphasize the need for a common content and navigation structure of course material while at

[3] The mentioned requirements are not only applicable to teachware but also to knowledge management systems in general.

the same time they call for domain-specific instantiations. This suggests to use a meta-modeling approach to hypermedia-based teaching material rather than to focus on a single model of course material.

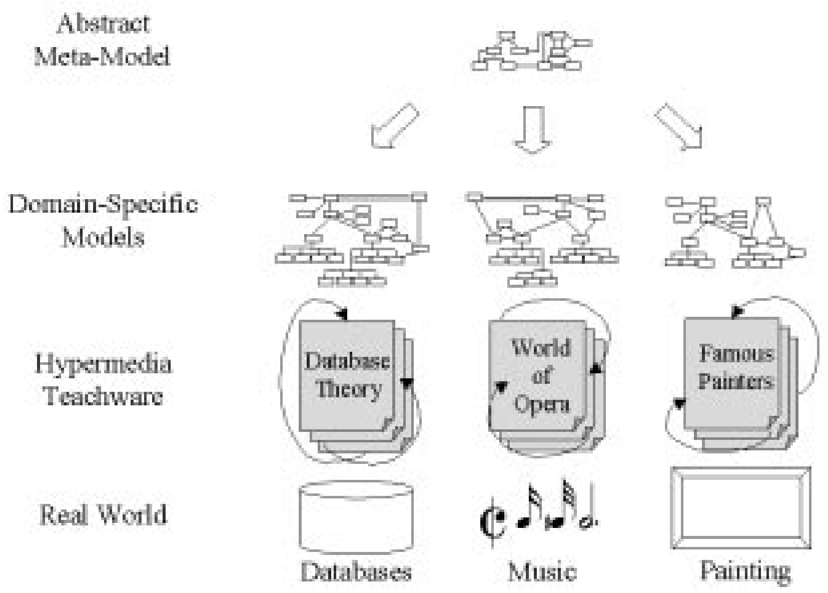


Fig. 1. Modeling at different levels

Figure 1 shows the different levels of our meta-modeling architecture:

1. At the bottom layer, the **real world** consists of the subjects to be taught or learned, or, in general, the knowledge to be managed. Two (obviously quite heterogenous) sample domains we refer to in this paper are database theory and the world of opera.
2. At the second layer, **hypermedia teachware** or, in general, hypermedia documents, including access and navigation aspects, are describing the given domain of application in an appropriate way using the domain-specific means of section 2.1.
3. In **domain-specific models**, we describe the content and navigation structure of course material or, in general, of hyperdocuments, of a given domain of application. Rather than modeling the topic itself - which is left to the *content* of the teaching documents - at this layer schemata are defined that control the admissible *form* of the documents.
4. Finally, the common **abstract meta-model** describes what it means to be a domain-specific model, i.e. gives a definition of the general kind of structure description that is accepted and can be understood.

3.2 The Abstract Meta-Model for Teaching Material

The properties found to be common to teaching material of different domains of application (cf. section 2.1) are realized in a straightforward way in the abstract meta-model, which we use to describe teaching material in general.

The graphical Unified Modeling Language (UML) [5, 8] is used to visualize our models. Note, that within our models, the concepts can be organized in a concept hierarchy with single inheritance relations among concepts. This generalization within the models is not to be mistaken with the generalization between the meta-model and a domain-specific model! In addition, concepts in both models which generalize other concepts are called abstract and cannot be instantiated. Especially, all concepts in the abstract meta-model are abstract.

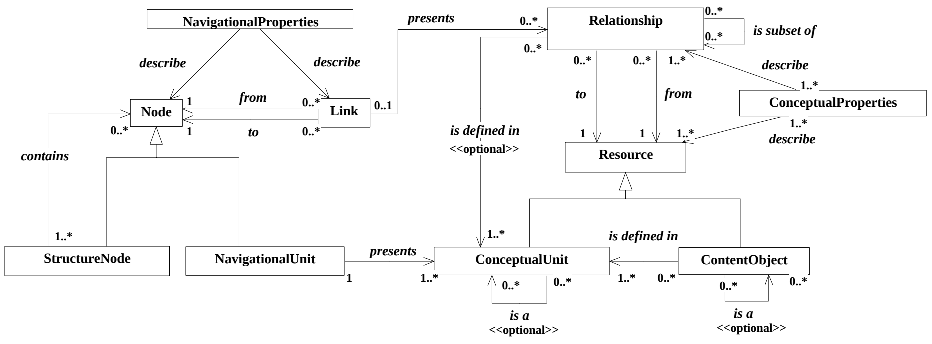


Fig. 2. abstract meta-model of teachware

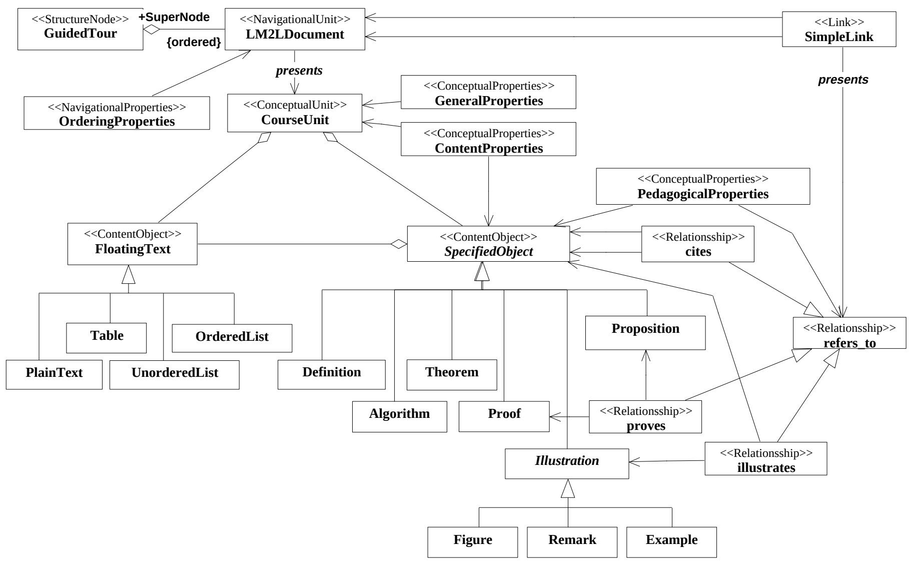
On the right hand side of figure 2, the abstract conceptual content structure of the course material is specified, i.e. which concepts model the content of teaching material. The conceptual units the teaching material consists of, are generalized to the abstract concept ≪*ConceptualUnit*≫. Their variable inner structure is realized by ≪*ContentObjects*≫. Both are, at the meta-model level, instances of the general concept ≪*Resource*≫ thus allowing all kinds of relationships, described by ≪*Relationship*≫, to hold for ≪*ConceptualUnits*≫ and ≪*ContentObjects*≫ as well. To allow the assignment of different types of meta-data to ≪*ConceptualUnits*≫, ≪*ContentObjects*≫ as well as ≪*Relationships*≫, we use the concept ≪*ConceptualProperties*≫.

On the left hand side of figure 2, the abstract navigational structure of teaching material is specified, reflecting the properties mentioned in section 2.1. ≪*ConceptualUnits*≫ are presented to the user by different kinds of ≪*Navigational-Units*≫ which are the terminal nodes of a polyhierarchical hypermedia structure similar to directories or books, but allowing nodes to belong to more than one supernode. Indeed, each ≪*Node*≫ has to be contained in at least one ≪*StructureNode*≫ allowing e.g. hierarchical as well as sequential navigation (as illustrated in section 3.3). The meta-model also specifies which ≪*Relationships*≫ are relevant to navigation and therefore are presented by ≪*Links*≫ which can be instantiated with different kinds of operational behaviour. To allow the assignment of different types of metadata to ≪*NavigationalUnits*≫, ≪*StructureNodes*≫ as well as ≪*Links*≫, we use the concept ≪*NavigationalProperties*≫.

3.3 Supporting Designers, Authors and Learners

This section discusses how designers of domain-specific models, authors, and learners can be supported e.g. by the exploitation of domain-specific inner structure as well as of metadata by a teachware management system.

Sample Domain-Specific Model As a sample instantiation of our meta-model, figure 3 shows a model[4] of course material on the domain of learning and teaching computer science which is based on real life course material.



<<StructureNode>> GuidedTour
+SuperNode
{ordered}
<<NavigationalUnit>> LM2LDocument
<<Link>> SimpleLink
presents
presents
<<NavigationalProperties>> OrderingProperties
<<ConceptualUnit>> CourseUnit
<<ConceptualProperties>> GeneralProperties
<<ConceptualProperties>> ContentProperties
<<ConceptualProperties>> PedagogicalProperties
<<ContentObject>> FloatingText
<<ContentObject>> SpecifiedObject
<<Relationsship>> cites
Table
OrderedList
PlainText
UnorderedList
Definition
Theorem
Proposition
<<Relationsship>> refers_to
Algorithm
Proof
<<Relationsship>> proves
Illustration
<<Relationsship>> illustrates
Figure
Remark
Example

Fig. 3. Domain-specifc model for computer science material

A domain-specific model defines in which instances of ≪*ConceptualUnit*≫ which instances of ≪*ContentObject*≫ can be present. In this model there is only one conceptual unit, namely *CourseUnit*, which can contain the following content objects: floating text, e.g. *PlainTexts*, *UnorderedLists*, *OrderedLists*, and *Tables*,

[4] Note that we have omitted names, roles, and multiplicities to increase readability.

or specified objects, e.g. *Definition*, *Proposition*, *Proof*, *Algorithm*, etc. Specified Objects can include floating text, e.g. a *Definition* can include an *OrderedList*, restricted by certain constraints like: a *Proof* must not contain another *Proof.*

Also instances of ≪*Relationship*≫ as well as their possible source and target objects are specified: Some of the content objects are connected by different instances of the conceptual relationship *refers_to.* For example, in the course material instances of all objects can be illustrated by a figure via *illustrates* and there are relations like *Proof proves Proposition.* On the one hand, as the relationship *proves* inherits from the relationship *refers_to*, the set of all instances of the *proves* relationship object is a subset of the set of all instances of *refers_to.* On the other hand, the relationship *proves* has as source the object *Proof* and as possible target the object *Proposition.* As inheritance between relationship objects also restricts the target and source objects of the inheriting relationship objects, the *proves* relationship only can start and end at objects corresponding to the definition of *refers_to* or subtypes thereof.

In our sample model there are three kinds of properties, i.e. groups of meta-data, describing *CourseUnits*, *ContentObjects* and *Relationships.* For the sake of simplicity, in our example, those properties are vectors of attribute/value pairs, where attributes are named properties of objects, and values are atomic (text strings, numbers, etc.). The properties in our example describe general aspects (*author*, *date*, *discipline*, and *language*), content aspects (*title* and *topics*) and pedagogical aspects (*difficulty*).

Finally, looking at the navigational aspect in our example, a *CourseUnit* on computer science (instance of ≪*ConceptualUnit*≫) is presented by a *LM^2L document* (see section 4 for details), which is an instance of ≪*NavigationalUnit*≫. These documents are grouped to sequences (*GuidedTours*) satisfying the partial ordering imposed by the two *OrderingProperties prerequesites* and *objectives*, which have as values text strings.

By presenting different types of relationships in the content model by different types of links in the navigational model, the navigational behaviour of relationships is separated from its conceptual meaning. It even is possible e.g. to specify an *is_prerequisite* relationship which is not navigable at all but can be used to compute the prerequisites of a certain learning goal which then are linearized and grouped to a *GuidedTour* to reduce disorientation of students.

Filtering Content Objects and Relationhips As we have modeled the inner structure of course units it is possible to filter content objects by type. For instance, when reusing existing material, it is possible, e.g. to ask for all *Examples* or to filter by subtopic using the *title* attribute which is contained in the content properties of any content object. Similarly, relationships can be filtered by type, which allows e.g. to ask for relationships of type *proves* only.

Adaptable Navigation and Presentation Relying on the filtering capabilities provided by the properties attributes of all objects, we are able to adapt the navigation through and the presentation of given contents and structures at the very fine granularity of single content objects. As an example, students of CS1 are shown exactly those *Definitions* which are appropriate to the knowledge level of beginners selecting according to an attribute *difficulty*.

Convenient Navigation In our sample model, there is a special access structure *GuidedTour* which supports sequential browsing of selected topics as suggested by the author. There may, of course, be several guided tours each of them adapting course material to a particular audience and aiming at a certain learning goal. Besides the navigational operations *next* and *previous*, authors and learners can also use the conceptual relationship structure as an ancillary structure, e.g. to navigate from a *Proof* via the *proves*-relationship to the corresponding *Proposition*.

Complex Query Capabilities The conceptual relationship structure mentionend above can also be used by a teachware management system to pose queries like "Which definitions and formal results are needed by undergraduate students of computer science to understand B-trees from an application point of view?"

Integrating New Material A *CourseUnit* as an instance of ≪*ConceptualUnit*≫ (see Figure 2) presents a self-contained sequence or set of content objects. The author can create a new course unit directly by creating a new LM^2L document (cf. section 4). She also can reuse existing material, e.g. a MS Word document. If this material has sectionings, an import wizard of a teachware management system could create course units using conventions such as 'use sections of highest sectioning depth as course units'. Furthermore, if there are specified objects, too, other conventions could be used, as well, e.g. 'group a proposition and the proof which proves it on the same course unit'. Thus, the author has full control over the size of course units (she could even define the entire course material, e.g. a big MS Word document, to be one course unit) while getting support for fine granularity modularization. Finally, if there is no inner structure at all, existing material can be integrated using wrappers, i.e. integrating it as an atomic course unit.

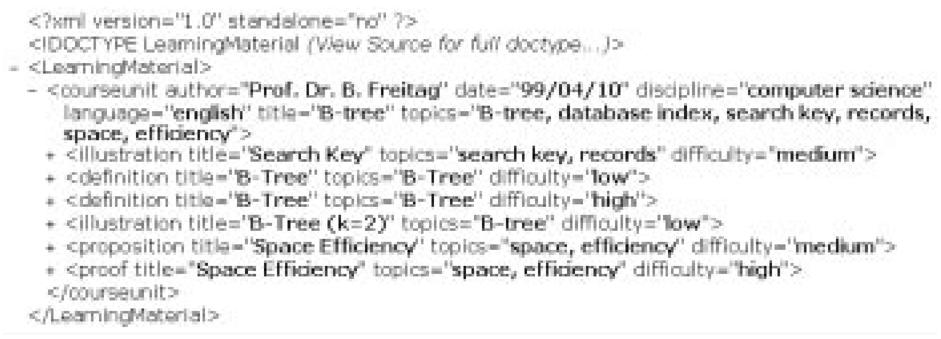
Combination and Sharing of Existing Material The use of metadata as described above allows authors to retrieve existing material according to various specifications and thus facilitates reuse and recombination. A teachware management system can use appropriate attributes for source code control as well as version control. In addition, even 'similar' material can be found, e.g. with similar pedagogical properties, provided that an appropriate metric is available. Analogously, metadata support sharing of material among several teachers.

Supporting Designers of Models As the abstract meta-model has been implemented as a generic framework, it is easy for designers to reuse and extend existing domain-specific models or to create new ones.

Suppose we want to integrate video clips into our course material on database theory, e.g. showing the teacher explaining an algorithm. Of course, multimedia objects like *animations*, *video clips*, or *audio samples* can be represented as content objects. All we have to do is to derive from the given abstract object *Illustration* a new object *TeacherVideo*. As an instance of *Illustration* it becomes a new possible source of the relationship *illustrates*.

To extend our framework we sketch how to design a completely new model for a different application domain. Suppose we want to describe learning material on the world of opera: In the appropriate content model there are no elements like *Proposition*, *Proof*, and so on. Rather we want to introduce concepts like *MusicScore* or *WorkDescription*. Thus, we define a new model for music material using these and other appropriate concepts but also reusing elements like *FloatingText* from our sample model for computer science material.

4 Learning Material Markup Language



```
  <?xml version="1.0" standalone="no" ?>
  <!DOCTYPE LearningMaterial (View Source for full doctype...)>
- <LearningMaterial>
  - <courseunit author="Prof. Dr. B. Freitag" date="99/04/10" discipline="computer science"
      language="english" title="B-tree" topics="B-tree, database index, search key, records,
      space, efficiency">
    + <illustration title="Search Key" topics="search key, records" difficulty="medium">
    + <definition title="B-Tree" topics="B-Tree" difficulty="low">
    + <definition title="B-Tree" topics="B-Tree" difficulty="high">
    + <illustration title="B-Tree (k=2)" topics="B-tree" difficulty="low">
    + <proposition title="Space Efficiency" topics="space, efficiency" difficulty="medium">
    + <proof title="Space Efficiency" topics="space, efficiency" difficulty="high">
    </courseunit>
  </LearningMaterial>
```

Fig. 4. Default view of B-tree.xml with collapsed content objects

In this section, we describe some implementation issues that make use of the advantages of our meta-modeling approach presented in the previous section. First, we need a format, in which structured learning material not only is stored at learning material web sites, but also can be easily accessed by teachware management systems.

As described in section 3.3, the conceptual units of our sample database teaching material are presented as LM^2L documents. LM^2L stands for Learning Material Markup Language, a XML application we have developed to markup formal teaching material in a Mathematics-oriented style.

The eXtensible Markup Language (XML) [16] is a restricted form of SGML retaining the power and flexibility of SGML while removing some complex features. XML and its family of technologies are standards managed by the World Wide Web Consortium (W3C) for representing information as structured documents.

The LM^2L documents contain marked up sections, so called elements, that represent conceptual content objects by syntactical means. For example, in course material on database theory, content objects like *Illustrations*, *Definitions*, *Propositions*, or *Proofs* describe e.g. B-trees in a Mathematics-oriented style. Figure 4 shows a MS Internet Explorer 5 default view of a sample LM^2L source file with collapsed (and marked with a plus) elements representing content objects.

Elements are declared in a Document Type Definition (DTD) [13], which expresses the structure of a document and consists of tag definitions and rules describing how elements may be nested, e.g.

```
<!--================= Document Structure ===============================-->
<!ELEMENT LearningMaterial (courseunit)>
<!ENTITY % specifiedobject
 "definition | proposition | theorem | algorithm | proof | illustration">
<!ELEMENT courseunit ( #PCDATA | %specifiedobject; | %floatingtext;)*>
```

To support adaptive presentation (see section 3.3) of course material and the transformation into different document formats if needed, e.g. HTML or LaTeX for web or paper publishing, we use the eXtensible Style Language (XSL) [16].

XSL is based on Document Style Semantics and Specification Language (DSSSL) and its online version, DSSSL-O, and also uses some of the style elements of Cascading Style Sheets (CSS) [16]. It is simpler than DSSSL, while retaining much of its power. As opposed to CSS, XSL can not only render a document and add structure to it but can also be used to completely rearrange the input document structure.

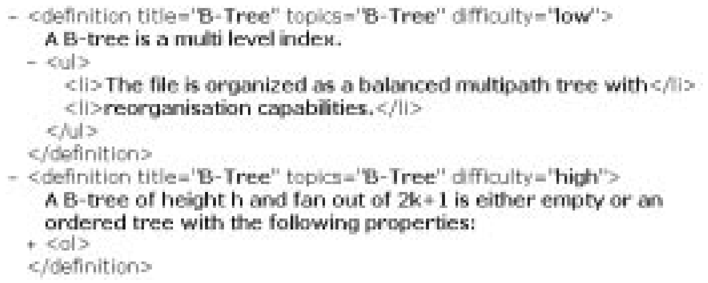
The elements in our material are enriched by attributes realizing the metadata proposed in section 3.3:

```
<!ENTITY % genattrs
 "author       %Text;             #REQUIRED
  date         %Date;             #REQUIRED
  language     %Text;             #IMPLIED
  discipline   %Text;             #IMPLIED">

<!ENTITY % contattrs
 "title        %Text;             #REQUIRED
  topics       %Text;             #IMPLIED">

<!ENTITY % pedagattrs
 "difficulty   (low|medium|high)  #IMPLIED">
```

XSL style sheets use these metadata, e.g. describing the different levels of difficulty (cf. figure 5), for restructuring and rendering the corresponding XML course material for different audiences, layouts or platforms in web based teaching providing adaptable presentation.



```
- <definition title="B-Tree" topics="B-Tree" difficulty="low">
    A B-tree is a multi level index.
  - <ul>
      <li>The file is organized as a balanced multipath tree with</li>
      <li>reorganisation capabilities.</li>
    </ul>
  </definition>
- <definition title="B-Tree" topics="B-Tree" difficulty="high">
    A B-tree of height h and fan out of 2k+1 is either empty or an
    ordered tree with the following properties:
  + <ol>
  </definition>
```

Fig. 5. Definitions of a B-tree with different levels of difficulty

5 Related Work

We discuss the distinction between conceptual and navigational aspects on two levels whereas e.g. [2, 9, 12] only use one modeling level. Work on hypermedia design like [14] and [4] also discusses presentational aspects, the former using a separate model for each aspect and the latter proposing a meta-modeling approach. In contrast to our work, the conceptual model of these approaches describes the domain of application presented by the given hypermedia documents.

The semantics of the real world is represented e.g. by entities and relationships in an extended ER notation [4] or by conceptual classes, relationships, attributes, etc. in an object oriented modeling approach [14].

[4] uses the Telos Language [10] for expressing a meta-model for hypermedia design. Like UML it offers a well-defined declarative semantics. For a subset of Telos modeling tools exist, too.

There are a number of (implemented) formalisms that are designed for conceptual modeling of the “meaning” of textual documents, e.g. Narrative Knowledge Representation Language (NKRL), Conceptual Graphs, Semantic Networks (SNePS), CYC, LILOG and Episodic Logic (for an overview, see [19]). As mentioned earlier, in our approach we refrain from representing natural language *semantics* but choose to model the *structure* of educational material.

[11] proposes knowledge items comparable to *CourseUnits*, but disregarding their inner structure.

Concerning the use of metadata, nodes, or node types in Computer Based Instruction (CBI) frameworks like [7] are described by a fixed set of attributes, e.g. *classification* (glossary node, question node), *control over access* (system, coactive, proactive), *display orientation* (frame, window), and *presentation* (static, dynamic, interactive). In our approach, the classification attributes could be instances of ≪*ConceptualUnit*≫, whereas the latter sets of attributes would belong to the corresponding properties. The types of links are also fixed: contextual, referential, detour annotational, return and terminal. Using a free definable domain-specific model, we are also not tied to a fixed number of structure nodes like *structure*, *sequence*, or *exploration* as in [3].

To organize metadata in the sample domain-specific model, we restricted ourselves to vectors of attribute/value pairs. In general, our abstract framework allows the use of arbitrary specifications of metadata, e.g. the use of labeled directed acyclic graphs which constitute the Resource Description Framework (RDF) model [17].

In the IMS approach [6] attributes similar to those used in our sample model are proposed which describe all kinds of educational resources in general and as a whole. Beyond that, in our approach, we propose domain specific metadata describing small parts of conceptual units, e.g. *Definitions*, *Proofs*, etc.

The XML based representation of teaching material developed in the TARGETEAM project [15] provides only a small fixed set of domain-independent content objects without attributes.

6 Conclusion and Future Work

A meta-modeling approach to adaptive hypermedia-based electronic teaching material has been presented which allows to describe knowledge about aspects of the contents of course material as well as navigational aspects. As an instance of the generic abstract meta-model we have described a sample computer science specific model for course material on database theory. Our approach combines some of the advantages of existing proposals and supports authors in adapting existing material to new audiences as well as learners in adapting content and navigation to their needs. As a foundation for future implementation we introduced the Learning Material Markup Language (LM^2L) to syntactically represent (formal) course material.

The techniques described in this paper are not only applicable to learning material but could also be applied to completely different types of knowledge material on the web.

Future work will consider the modeling of other domains of applications as well as the corresponding generalizations to LM^2L needed to make it suitable to markup a wider variety of course material by using modularisation. Thereby, we are also integrating SMIL [18] to describe multimedia contents as well.

Finally, a web based teachware management system is about to be implemented which comprises the meta-model introduced in this paper as well as methods to generate and verify the models governed by it. It will also provide tools to support the creation and management of teachware, especially the composition and configuration to new audiences and learning goals.

References

1. S. Abiteboul, S. Cluet, V. Christophide, M. Tova, G. Moerkotte, and J. Siméon. Querying Documents in Object Databases. *International Journal on Digital Libraries*, 1(1):5–19, 9 1997.
2. P. Brusilovsky. Efficient Techniques for Adaptive Hypermedia. In C. Nicolas and J. Mayfield, editors, *Intelligent hypertext: advanced techniques for the World Wide Web*, volume 1326 of *LNCS*, pages 12–30. Springer, Berlin Heidelberg, 1997.

3. M. Casanova, L. Tucherman, L. M., J. Netto, N. Rodriguez, and G. Soares. The nested context model for hyperdocuments. In *Hypertext '91 Proceedings*, pages 193–201. ACM, ACM Press, 1991.
4. P. Fröhlich, N. Henze, and W. Nejdl. Meta-Modeling for Hypermedia Design. In *Proc. of Second IEEE Metadata Conference, 16 - 17 Sep. 1997, Maryland*, 1997.
5. B. Grady, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Object Technology Series. Addison-Wesley, 1999.
6. IMS. Instructional Management Systems Project. http://www.imsproject.org/.
7. P. Kotze. Why the Hypermedia Model is Inadequate for Computer-Based Instruction. In *SIGCSE 6th Anual Conf. on Teaching of Computing, ITiCSE'98, Dublin, Ireland*, volume 30, pages 148–152. ACM, ACM, 1998.
8. C. Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Addison-Wesley, 1997.
9. J. Mayfield. Two-Level Models of Hypertext. In C. Nicolas and J. Mayfield, editors, *Intelligent hypertext: advanced techniques for the World Wide Web*, volume 1326 of *LNCS*, pages 90–108. Springer, Berlin Heidelberg, 1997.
10. J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing Knowledge About Information Systems. *ACM Transactions on Information Systems (TOIS)*, 8(4):325–362, 10 1990.
11. W. Nejdl and M. Wolpers. KBS Hyperbook – A Data-Driven Information System on the Web. In *Proc. Of WWW8 Conference, Toronto, May 1999*, 1999.
12. L. Rutledge, L. Hardman, J. v. Ossenbruggen, and D. C. Bulterman. Structural Distinctions Between Hypermedia Storage and Presentation. In *Proc. ACM Multimedia 98, Bristol, UK*, pages 145–150. ACM, ACM Press, 1998.
13. Ch. Süß. Learning Material Markup Language.
http://daisy.fmi.uni-passau.de/PaKMaS/LM2L/.
14. D. Schwabe and G. Rossi. Developing Hypermedia Applications using OOHDM. In *Electronic Proceedings of the 1st Workshop on Hypermedia Development, in conjunction with 9th ACM Conf Hypertext and Hypermedia, Pittsburgh, USA, 1998*, Hypermedia Development Workshop. ACM, 1998.
15. G. Teege. Targeted Reuse and Generation of TEAching Materials.
http://www11.informatik.tu-muenchen.de/proj/targeteam/.
16. WebConsortium. eXtensible Markup Language. http://www.w3.org/XML/.
17. WebConsortium. Resource Description Framework. http://www.w3.org/RDF/.
18. WebConsortium. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. http://www.w3.org/TR/REC-smil/.
19. G. P. Zarri. Conceptual Modelling of the "Meaning" of Textual Narrative Documents. In *Foundations of Intelligent Systems, 10th International Symposium, ISMIS '97, Charlotte, North Carolina, USA, October 15-18, 1997, Proceedings.*, volume 1325 of *LNCS*, pages 550–559. Springer, 1997.