Abstract. The use of adaptive hyperbooks can considerably improve the quality of distance education. However, their design and implementation is a resource-intensive and complex task. Although object-oriented methods for hypermedia design in general have been proposed, they do not cover all the aspects necessary for educational hyperbooks. In this paper we propose a framework for hyperbook design, which covers all aspects of a hyperbook model: Domain Modeling, Navigational Modeling, User Modeling, and Visualization. Furthermore, we present a system architecture, which uses a fully generic client/server application for displaying the hyperbook from a WWW server to a WWW client.

Introduction

In our KBS Virtual Classroom Project we are working on providing students with complete teaching and learning environments for several subjects of computer science (artificial intelligence, software engineering and introduction to programming). To encourage project-oriented education, students work in accompanying projects in each of these lectures. While current lecture notes are still based on ordinary texts, we have been working on a hyperbook for software engineering and programming, which tightly integrates case descriptions and solutions, concept descriptions and more conventional tutorial notes etc. As this hyperbook shall be extendible by different authors as well as by students, a systematic model of all aspects of the hyperbook (domain structure, navigation, visualization and adaptation) is necessary.

As a general discussion of educational concepts is not important for this paper, we simply refer the reader to [6] for a general discussion of concepts, benefits and costs of constructivist models of teaching and learning and to several specific uses of such an approach e.g. [16] and [7].

In the last years, several methodologies for hypermedia modeling have been developed, among which the RMM method [9] and the OOHDM method [17] are the most prominent examples. These methods provide primitives for modeling a hypermedia application domain, based on ER diagrams (RMM) or Rumbaugh's OMT method [15] (OOHDM). Based on this domain model the possibilities for navigation are described. As pointed out in [9] such methods are best suited for designing front ends to loosely structured data. Therefore, the navigational concepts proposed in these methods are geared towards indexing a large amount of relatively simple information pieces. On the other hand, education hyperbooks have to include various forms of (sequential) tours, user models to adapt to the user's experience and visualization models. Furthermore, the system should support the definition and storage of these models as well as the hyperbook itself, and generate hyperbooks dynamically based on these models.

On the other hand, current work in adaptive hypermedia textbooks (such as [11], [2], [4], [3]) concentrates on building domain models and user models for adaptive hyperbooks, using semantic nets to describe these domain models and to index the hyperbook with the corresponding domain nodes. We build on these concepts and focus on the modeling language and meta-language for building domain models, explicit navigation and visualization models and user models. Other related work includes extended hypermedia editors and environments (such as [8], [18]), which do not focus on domain or user modeling, and the hyperbook by Schank and Cleary [16], which uses a set of typed links, but no domain or user model.

Data Modeling Language

For the declarative representation of the hyperbook data models we use a dialect of the object-oriented conceptual modeling language Telos [13], which is implemented in the ConceptBase system [10]. This language combines object-oriented concepts with deductive rules and constraints. Due to its representational power Telos is suitable for meta modeling, i.e. for describing domain-specific modeling languages (e.g. [14]). In this spirit, we have used a Telos meta model to define the primitives used for domain modeling, navigational modeling, user modeling and visualization modeling, as shown in figure 1. In this paper we will however not discuss the formal details of our Telos model. Instead, we will introduce the various concepts of our modeling language informally on the basis of examples.

Domain Model

The domain model describes the entities of the application domain and their relationships. In our framework the domain model is
expressed by Domain Classes and Relationships. Domain classes are characterized by their attributes. The domain classes for an application can be organized in a class hierarchy with single inheritance relations among classes. We support binary relationships between domain classes, which are classified by their cardinality (1:1, 1:n, m:n). In addition we allow to specify a key for the n-side of a 1:n relationship. As shown in figure 2 our notation for these modeling concepts is an extension of Entity Relationship Diagrams. Figure 3 provides a small example, taken from our software engineering book: A Software Process consists of several Phases, thus there is a 1:n relationship between Software Process and Phase. This relationship is indexed by the key attribute PhaseNr (Phase Number). As we will see in section 2.2, this key attribute specifies the order in which the Phases are presented. At the bottom of figure 3 we see the Waterfall Model as an instance of Software Process which is related to its phases.

Figure 2: Domain Modeling Notation

Our domain modeling technique is similar to RMM but extends it by inheritance and m:n-Relationships. The possibility to specify a key for the n-side of a 1:n-relationship is also an extension. In contrast to the semantic models used by our method and RMM, Schank and Cleary do not use an explicit domain model; they only divide the links among the pages into certain types.

Navigational Model

Data Model Navigation

The relationships defined in the domain model are the basis for systematic semantics-based navigation in the book. As in RMM, this type of navigation is defined on the schema level, i.e. for every navigational construct, it is defined, which relationship it supports. Database queries are used to obtain the actual hyperlinks. This enhances the extendibility of the document since the link structure automatically adapts to the database content.

Figure 3: A Simple Domain Model. In the upper part of the image the meta model concepts are shown. (full-size)

Links support bidirectional navigation based on 1:1-relationships. Links are displayed by inserting a hyperlink to the corresponding domain object in each page. Following the RMM method, we provide three concepts for navigating a 1:n-relationship: An Index resembles a table of contents: It consists of a list of links to related objects. If an index key is specified for the relation, it determines the order of the links. In the example from figure 3 links to the phases Requirements, Design, Implementation and Integration are inserted in
the Waterfall Model page. Since Requirements has the smallest key the corresponding link appears first. A **Guided Tour** is an alternative representation of an 1:n-relationship. Here, a link to the first phase is inserted in the Waterfall Model page. Each phase page contains a link to its successor (the phase with the next larger key). The last phase page links back to the Waterfall Model page. An **Index Guided Tour** combines the previous concepts: Each page has links to the next in a sequence and also to the main page (the Waterfall Model page in our example). We extend RMM's navigational constructs by a **Cross Reference Index** concept, which we find suitable for our programming books. This concept allows to navigate a m:n-relationship. For example, it is possible to navigate from a program to programming concepts used and from there to other programs using the same concepts.

**Sequential Navigation**

We stress that data model-based navigation allows to access information more selectively compared to a conventional book. However, the possibility to read parts of the book or the whole book sequentially provides important guidance especially for the beginner. Therefore we define the concept of a **Reading Sequence**. A reading sequence consists of a sequence of pages connected by hyperlinks, whose graphical representation differs from the data model links shown above (reading sequence links are shown as 3D buttons at the top of the page while data model links are shown as usual blue hyperlinks at the end of the page). Our framework allows the automatic construction such reading sequences from the domain model structure.

![Figure 4: Database representation of the internal structure of a page](full-size)

**Visualization Model**

Up to now we have modeled the concepts of the application domain and their relationships (domain model) as well as the possibilities to navigate through these topics (navigation model). The next step is specify how information is presented. In our system, topics are presented by **WWW Pages**. A WWW Page itself consist of a sequence of fragments containing **MIME Objects**. **MIME** [1, 12] (Multipurpose Internet Mail Extension) is a standard for sending different types of content (Text, Pictures, Sound, Video) over the Internet. MIME is supported by most EMail and WWW clients. The MIME type hierarchy is part of our modeling language. Each MIME Object belongs to a MIME Type, which tells the hyper book server how to insert it into the WWW Page and tells the hyper book client how to display it (see section 3). The connection between the Visualization Model and the Domain Model is that every WWW Page, i.e. collection of MIME Objects **implements** a domain object. To see the practical impact of these definitions, consider another software engineering example shown in figure 4. The domain object **Use Case** is an instance of the domain class **Method**. It is implemented by the WWW Page **Use Case Page**. This page consists of 3 fragments: The first contains a document of type TEXT/HTML providing an overview of Use Cases (use_case_overview.txt), the second contains a picture of type IMAGE/GIF, which gives an example of a use case diagram and the third is another document of type TEXT/HTML describing the impact of use cases for the design phase.

**User Model**

We use a simple user model to adapt both page content and navigation structure of hyper books. This model consists of two parts:

**Model of general prerequisite knowledge:**

When using the hyper book for the first time, the user has to fill in a form where he judges his background knowledge in topics related to the book on a numerical scale. For the software engineering book the general background knowledge consists of other topics in computer science like programming language experience or database experience. This general experience is represented in the database by attributes of the user object.

**Model of learning progress:**
Here we use a simple form of overlay modeling [5]. Each domain object is regarded as a topic, on which the user gains experience by reading the corresponding WWW page. Thus, the database keeps track of the pages the user has read.

The adaptation of content based on this information is achieved by constraints on the visibility of fragments. In ConceptBase constraints are expressed in a variant of typed first-order logic. This allows to express complex conditions declaratively. Consider a page on interface design, which contains as an example an abstract data type for sets in C++. The fragment containing the example shall only be displayed if the user has good knowledge in C++ (rated on a scale from 1=no knowledge to 5=perfect knowledge) and has read the page on abstract data types. This is expressed as

\[
\forall u/\text{User} (\text{this visibleTo } u) \Rightarrow ((u \text{ hasRead ADT_Page}) \text{ and } \exists \text{ rating/Integer } (u \text{ cppKnowledge rating}) \text{ and rating } > 3)
\]

This constraint states that the fragment (this) is only visible to users \( u \), who have read the page called \( \text{ADT_Page} \) and whose value of the \( \text{cppKnowledge} \) attribute is greater than 3. The adaptation of navigation works in a similar fashion. The navigational objects (links, indexes, ...) are augmented with constraints on their visible instances. Our current adaptation model can be seen as a database-oriented reconstruction of Kay and Kummerfield's customization technique [11], which controls the visibility of texts by preprocessor statements. We consider our modeling technique more declarative, because visibility conditions and the internal structure of the pages are explicitly represented as part of the data model.

**System Architecture**

In the previous sections we have outlined how all data relevant to the hyper book is stored in ConceptBase following a declarative meta model. Figure 5 shows the basic system architecture, which is based on standard components complemented by a server-side applet and the ConceptBase database manager. After logging into the system by activating the login applet a user-specific starting page is displayed. The user navigates the hyper book by activating links. These links however do not represent static HTML pages but start CGI scripts instead. Whenever a hyper link is activated the name of the corresponding domain object plus the name of the user are passed to a server-side applet. The applet queries the data base for the fragments of the page representing the domain object and for the domain object's navigational possibilities. From this information it constructs a user specific page. It does so by combining the visible fragments in a HTML page. Then it updates the user model by marking this page as read and uploads the page on the server. The page is then displayed to the user in his WWW client.

**Figure 5:** System Architecture

**Conclusion and Future Work**

We have discussed a methodology and modeling language for describing all aspects of hyperbooks, including a domain model, navigational model, visualization model and user model. These models explicitly specify all design decisions needed for adaptive hyperbooks, and are also the basis for automated creation of customized pages which are displayed using a standard WWW client/server architecture. Hyperbooks modeled following our framework can be navigated both sequentially and systematically based on the domain structure. The user model still has its limitations. Our current model is based on the fact that a user visits pages. It cannot capture how intensively he has perused their content. Therefore, a future challenge is to add exercises and tests to improve our user model.

**References**


About this document ...


The command line arguments were:

```
latex2html -split 0 -no_navigation ahh.tex
```

The translation was initiated by Peter Froehlich on Mon Jul 14 13:53:35 MET DST 1997

www@kbs.uni-hannover.de

Last modified on July 14, 1997

Motivation: Hyperbooks and Modeling Modeling Framework Meta Model Domain Model Navigational Model Visualization Model Previous Approaches Slideshow... Å Hyperbook Modeling extends modeling techniques (semantic data modeling, object oriented software models) by taking hypertext properties (navigation, visualization, etc.) into account. Modeling Framework. Explicit modeling of all hyperbook aspects. Å Definition of a systematic approach to hyperbook design containing. Domain Model. Navigation Model. The design model of an intelligent tutoring system is directed towards the separation between the domain knowledge and the tutor shell. The architecture is composed by a user model, a knowledge base, an adaptation module, a pedagogical module and a presentation module. Bayesian Networks are used to assess userâ€™s state of knowledge and preferences, in order to suggest pedagogicalâ€¦ CONTINUE READING. View Paper. Save to Library. Create Alert. Cite. Å A database-oriented approach to the design of educational hyperbooks. William M. McLaren, Wolfgang Nejdl. 1997.