International Journal fo Artificial Intellligence in Education (2001), to appear

Adaptation in Open Corpus Hypermedia

Nicola Henze and Wolfgang Nejdl
ITI-KBS, University of Hannover, Appelstr. 4, D-30167 Hannover, Germany
{henze, nejdl}@kbs.uni-hannover.de

Abstract: Adaptive hypermedia courseware benefits from being distributed over the Web: content can always be kept up-to-date, discussions and interactions between instructors and learners can be supported, new courses can easily be distributed to the students. Nevertheless, adaptive hypermedia systems are - even in the web context - still stand-alone systems as long as they lack the ability to integrate and adapt information from arbitrary places in the web. In this paper we discuss the concept and realization of an adaptive educational hypermedia system which facilitates adaptation of distributed learning materials. The proposed system is based on an indexing concept for various information resources and a Bayesian Network engine for calculating estimations on a user's changing knowledge. It implements constructivist teaching approaches, establishes conceptual models for designing courses and enables guided tours, intelligent example retrieval and further adaptive navigation support. Advantages and problems in designing adaptive open corpus hypermedia will be discussed in a concluding section.

INTRODUCTION

Since the emergence of the World Wide Web (WWW) in 1991, the value of information has got a new dimension. Nowadays, millions of computers are connected via the internet, humans can collect information from nearly anywhere in the world.

This enormous amount of information is also a chance for experience and learning. But effectively selecting information from the internet is still a hot research topic as the effectiveness of search machines increases with the precision of the query. The information contained in the internet is often useless for exploring or learning, as learners need guidance to build up a mental model of the area they are working on before being able to make sufficiently exact queries.

It would be very helpful to have different textbooks for different types of learners, for students with different interests and different initial knowledge on the topic and to have textbooks which can make advantageous use of the internet. To make a step in this direction, adaptive hyperbooks personalize the content of hyperbooks to the particular needs of users. They give users the ability to define their own learning goals, propose next reasonable learning steps to take, support project-based learning, give alternative views, and they can be extended by documents written by the learners. Adaptive hyperbooks are information repositories for accessing distributed information. Implemented as internet applications, they can integrate documents located anywhere in the web - and adapt these (distributed) documents to the learner's goals and knowledge.

In this paper we propose concept and realization of an adaptation component for an *educational hypermedia system* which, on the one hand, implements advanced teaching strategies, and, on the other hand, enables integration and adaptation of learning material found in the WWW i.e. works on an *open corpus*. We give a short introduction to constructivist learning in distance education and discuss how we assembled a constructivist learning environment in the KBS hyperbook system. The basic functionality of hyperbooks will be described. In the section on the adaptation component we describe the student modeling in hyperbooks including the "knowledge vector" which codes a student's actual knowledge, and the Bayesian network engine underlying the calculations for the knowledge vector. The following two sections show the use of metadata to facilitate adaptation of distributed learning

materials to a particular student's needs and the various adaptation functionalities of hyperbooks. A review of related work compares the proposed approach of hyperbooks to other approaches in adaptive educational hypermedia which focus on the same adaptational functionalities as the hyperbook system. In the concluding section we discuss advantages and problem in designing open adaptive hypermedia systems.

THE EDUCATIONAL APPROACH

Since hyperbooks are web applications, they are typically used in distance learning scenarios, where a learner uses the information from the hyperbook on his own. Thus, it is important to think about useful teaching strategies to encourage a learner to learn *actively* and not just to read or "consume" *passively* the information. For this purpose, we emphasize constructivist learning strategies, for example by integrating problems or "real world tasks" in the curriculum of a hyperbook, and by structuring the hyperbook based on projects and their relationship to information pages. Learners can reach learning goals or can receive answers to information requests while working on these problems, which introduce, explain and show the use of the learning items.

Constructivist Learning Approach in Distance Learning

Computational learning environments benefit from a strong background in educational theory. Simply reproducing conventional teaching and learning concepts in a computational environment does not utilize these new technologies. Educational models, which show particularly interesting features for many parts of academic education are *constructivist models of learning and teaching* (see also the discussion of the relationship between such models and the field of instructional technology in [Duffy and Jonassen, 1992]).

Virtual learning environments, if designed properly, will provide the functionality to support improved concepts in education theory which are difficult to realize without the help of new communication and networking technologies.

Critical elements in the design of constructivist learning environments are the specification and integration of authentic and complex activities during the learning process [Honebein et al., 1991]. In a constructivist environment students are encouraged to solve problems which could also occur in the real world. The learning environment simulates the context of the problems, on which the students perform these authentic activities: they have to decide how to structure and solve the problem, collect background information, develop solution strategies, etc. Authentic activities shift the responsibility for both selecting and performing tasks from the teacher to the learner. Students therefore use the course material actively and gain deeper insight. A project-based approach can be chosen to implement this constructivist learning model: the global project context stimulates the learning activity while the responsibility of the students to find a suitable solution strategy on their own leads to activities which increase the students' problem solving competence. The environment for such projects has to provide references, case-studies, background, and related information as well as a working environment for software projects carried out by the students parallel to the lecture. This environment has to reflect the user's knowledge and state of his work to present only appropriate information.

Case study: Implementing Constructivist Teaching Concepts in a CS1 Course

In the following we describe the realization of constructivist teaching concepts in a CS1 (Computer Science 1) course [Henze and Nejdl, 1997]. This CS1 course is an introductory course to programming in Java, given for undergraduate students of electrical engineering and computer science.

Based especially on the ideas of learning as design activity (as advocated by Papert and his colleagues [Papert, 1993, Kafai and Resnick, 1996]) and learning as an intentional activity involving knowledge-building and discussion (as in CSILE [Lamon et al., 1993, Hewitt and Scardamalia, 1996]), we focus in our CS1 course on the following two issues:

- Integrating goal-oriented learning and projects (authored by lecturers and students) into our course materials, and
- Connecting student projects with the rest of the course material showing which CS1 concepts have been applied (and thus learned) to which part of the project.

The conceptual model of our hyperbook concentrates on this problem-oriented and inquiry-oriented aspect, and explicitly models the relevant aspects to support a goal-directed and inquiry-oriented learning style. Students need to know which materials are necessary for specific projects, and can use personalized learning sequences and information indexes to retrieve the required information and hyperbook pages.

Goal orientation is an important aspect of our educational hyperbooks. Since we do not want to determine the learning path of a student or a student group from the beginning to the end, the students are free to define their own learning goals and their own learning sequence. In each step they can ask the hyperbook for relevant material, teaching sequences and hints for practice examples and projects. If they need advice to find their own learning path they can ask the hyperbook for the next suitable learning goal.

Discussion: Requirements for enabling project-based learning

It is a central requirement of a constructivist teaching approach to keep the hyperbook as maintainable and expandable as possible. A maintainable structure allows the integration of students' results seamlessly and to minimize the effort require to maintain the theoretical course materials. The implementation of such a sophisticated conceptual backbone requires rigorous modeling. We propose a *conceptual modeling* approach for hyperbooks which explicitly represents all aspects of the hyperbook application domain, and access to information.

Another important requirement is the information presentation in a project-based learning approach. Since the students are supposed to work on their course projects or on smaller projects or examples contained in the hyperbook, they require information and background knowledge. Therefore it is a task of the hyperbook *to select and present suitable information to a user*. For integrating student projects into the hyperbook, we use the idea of *portfolios*. The students demonstrate in their portfolio which concepts they have studied for their course project.

Since the students are working with the hyperbook in the internet, the integration of useful information present in the internet into the learning material is near at hand. The hyperbook should serve as an *up-to-date information repository*. By selecting and annotating the information according to the student's goals and knowledge it should support their project work. In addition, *reading sequences* must be generated. They lead the student - based on his actual knowledge - towards the requested information, including necessary prerequisites he actually needs to know. Since working on a larger project requires the definition of subprojects or subtasks which have to be solved first, the hyperbook must support goal-based learning. A user must be able to *define learning goals* on his own. Moreover the system must be able to generate appropriate learning goals and learning steps for the user. To reach such a (proposed or self defined) learning goal, the system should be able to find relevant projects and examples related to the goal. Therefore, selection algorithms must be found. They should present the most suitable projects to the user which match his current learning goal, considering his actual knowledge.

THE KBS HYPERBOOK SYSTEM

The KBS hyperbook system is a tool for modeling, organizing and maintaining adaptive, open copurs hypermedia on the WWW. *Open* in this context means that these hypermedia systems are able to integrate distributed information. The system has been developed at the Institut für Rechnergestützte Wissensverarbeitung, University of Hannover, whose English name "Knowledge Based Systems Group" (KBS), gave the name for the project.

In this section we will describe how we use a conceptual model for modeling course and instruction material in the hyperbook system by the example of the CS1 course, the earlier mentioned Computer Science 1 course (introduction to Java programming for undergraduate students). We show the way we integrate student projects into the hyperbook based on the idea of portfolios.

Conceptual Modeling for Adaptive Hyperbooks

The KBS hyperbook system structures and displays hypertext materials based on conceptual models. This section describes the conceptual model, which models courses, different kinds of materials (such as projects, examples, portfolios, HTML pages), and the integration of information from the World Wide Web. As an example we show the integration of the Sun Java tutorial [Campione and Wallrath, 1999] into the hyperbook. The Sun Java tutorial is free available on the internet and thus very suited for being integrated into the learning material of the CS1 hyperbook.

For the declarative representation of the hyperbook data models we use the object oriented conceptual modeling language O-Telos [Mylopoulos et al., 1990], which is implemented in the ConceptBase system [Jarke et al., 1995]. 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. [Nissen et al., 1996]). This allows us to research on meta modeling for the KBS hyperbook system [Nejdl and Wolpers, 1999,Nejdl et al., 2000] using e.g. ideas and concepts of the Resource Description Framework [W3C Working Group, 1998] (RDF).

Modeling courses and lectures

Central to this part of our conceptual model (see concepts highlighted in light grey in Figure 1) is the entity course which represents a real course given at some university or at other institutions. Each course consists of several lectures.

The course has a relation to the *course group* (Figure 1). A course group integrates different courses on the same topic. Take, for example, the CS1 course. In winter semester 1999 / 2000, the Institut für Rechnergestützte Wissensverarbeitung at the University of Hannover held this course for undergraduate students of electrical engineering and technical computer science. A similar course, with support of that institute, is given at the Freie Universität Bozen, Italy. Both CS1 courses are modeled as courses, and belong to the course group "CS1". Each course has its *glossary* (for the generation of the glossary entries, and a number of *areas* which structure the application domain.

The embedding of project and portfolios in the learning material is an important part of our teaching concept. To model the integration, each course is related to *projects* (Figure 1). These projects can be the *actual projects* of the course on which the students work. Or they can be *former projects*, which give examples of projects performed by students of past courses and contain the portfolios of them.

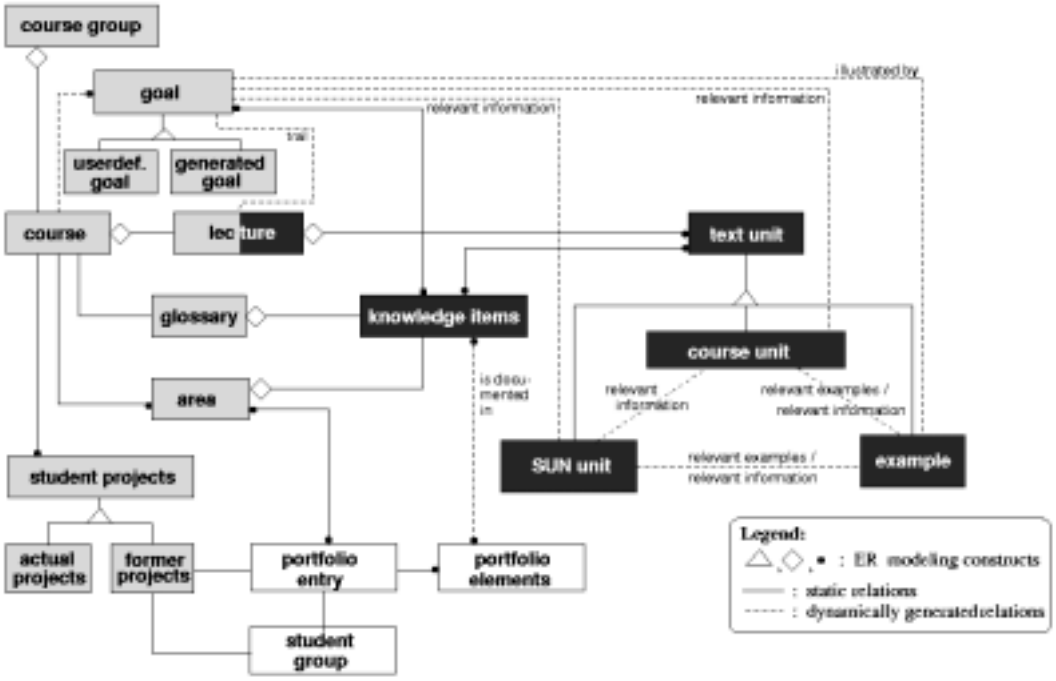

Figure 1: Conceptual model of the CS1 hyperbook, course related modeling is highlighted in light grey, modeling of different information resources is highlighted in dark grey. The relations displayed by dotted lines are dynamically generated at real-time and enable individual navigation of the content depending on a users current goal, knowledge, assignment, etc.

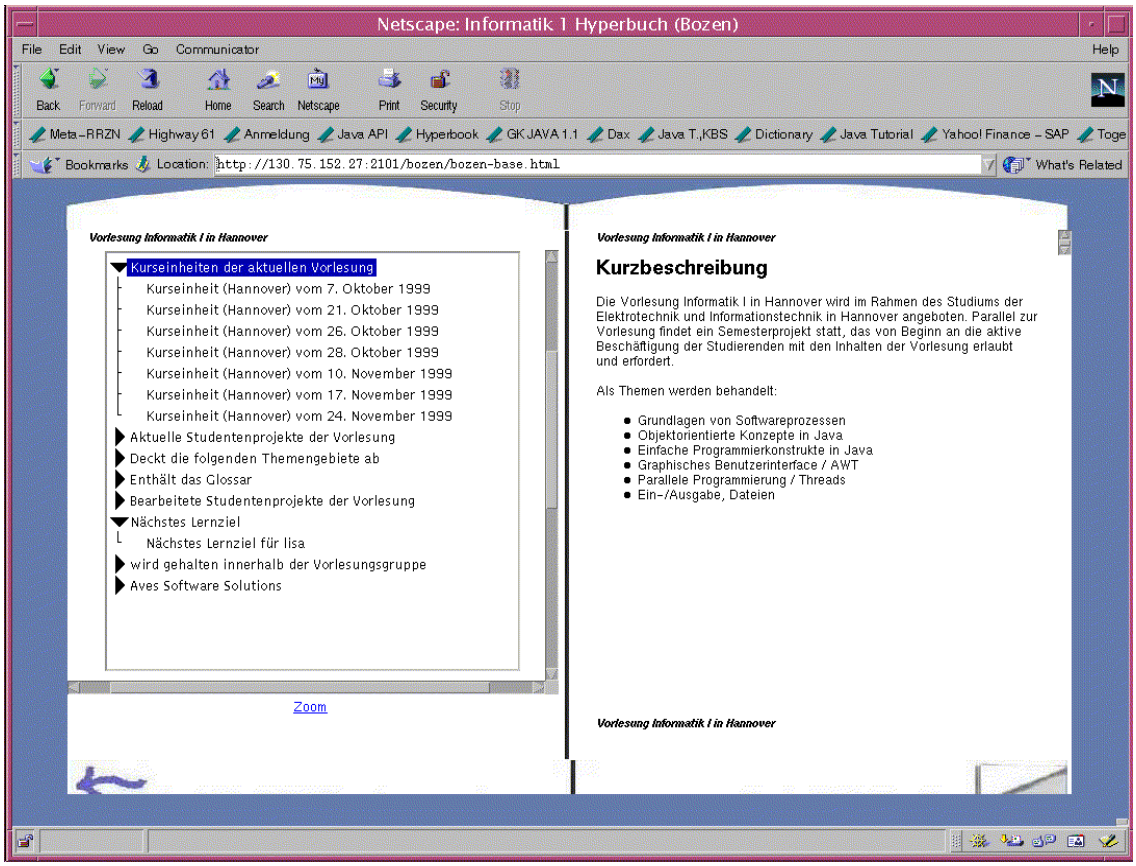

Figure 2: Example for a course modeled in the KBS hyperbook system

To support goal-oriented learning, students can define their own learning goals (*user defined goals*) or can request new reasonable goals from the hyperbook (*generated goals*). The selection and the support of goals will be discussed in the section “enabled adaptation”. Figure 2 gives an example of the CS1 course given in winter semester 1999 / 2000. Each above mentioned relation from a course to other concepts is processed by the KBS hyperbook system at real-time to a link in the left frame. Specific lectures of this course can be seen, current student projects, the areas of the domain of this course, examples of former projects, the reference to the next reasonable learning goal, and the reference to the lecture group.

Modeling Different Information Resources

Each lecture consists of a sequence of text *units* which are used by the teacher during the lecture (see the concepts highlighted in dark grey in Figure 1). A text unit can be a *course unit*, thus an information page belonging to the hyperbooks's library. Or it can be an *example* showing the use of some concept. As the KBS hyperbook system allows integration of information located anywhere in the WWW, these text units can also be information pages in the WWW, for example pages in the *Sun Java tutorial*.

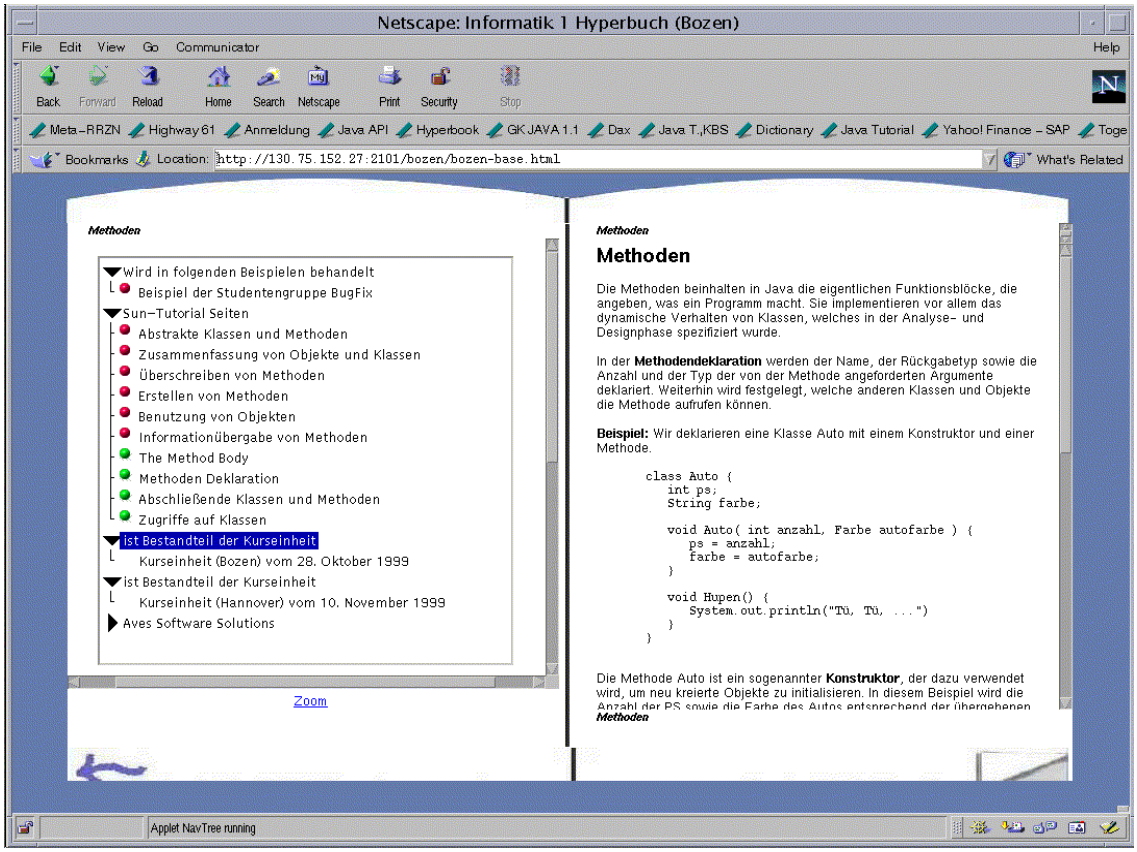

Figure 3: Course unit “Methoden” with dynamically generated links to examples, Sun Java tutorial pages and to the two lectures where it occurs.

From each types of these text units, links to related information are generated by the adaptation component (see section “enabled adaptation”). For example, from a course unit, links to relevant examples are generated, and links to relevant Sun tutorial pages, which give alternative descriptions of these concepts. In Figure 3, we see on the right hand side the HTML page about methods from the hyperbook library. The use of methods can be studied in the example of the student group “BugFix” (uppermost link on the left hand side), and the Sun Java tutorial contributes many links to relevant information pages. As the course unit “Methoden” is contained in a lecture, we also find a link to this lecture.

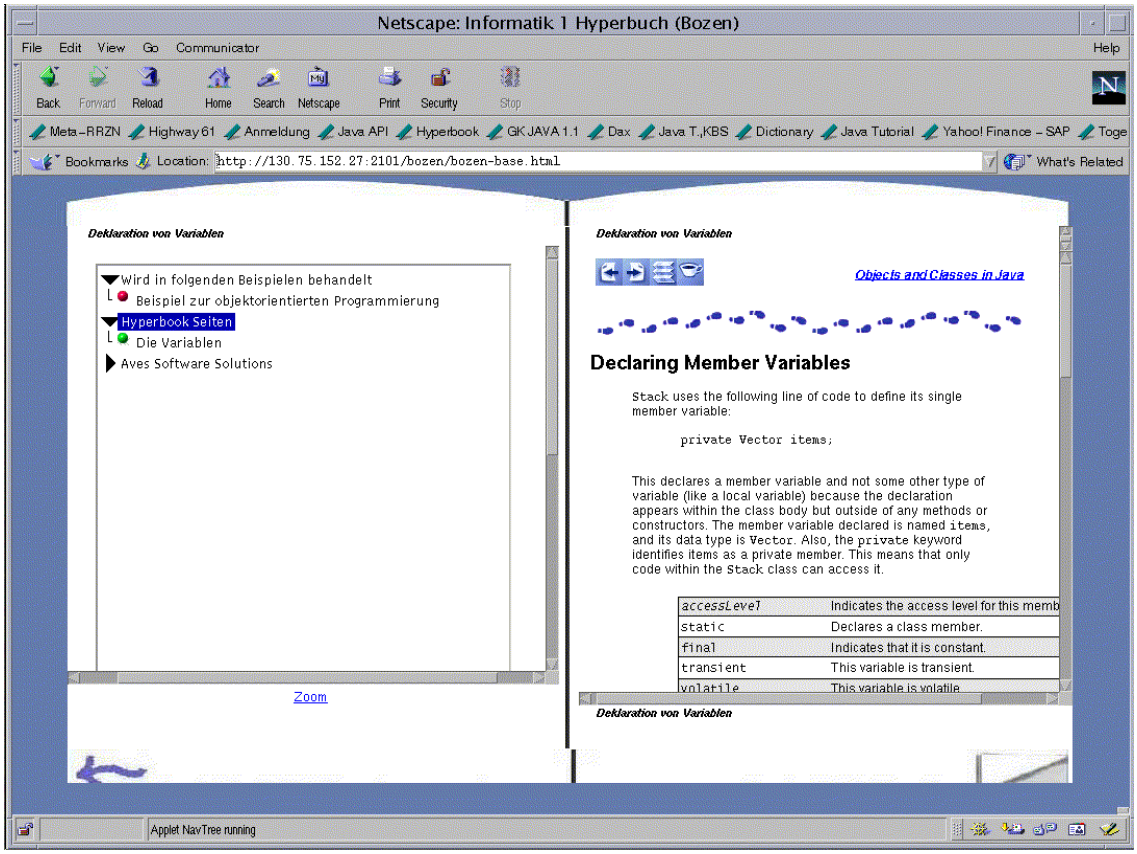

Figure 4: Example of the integration of Sun Java tutorial pages in the KBS hyperbook system

In Figure 4, we see the integration of a Sun Java tutorial page into the hyperbook. The page itself is displayed in the same manner as if it originated from the hyperbook library. We stream such pages without any modifications into the hyperbook. Thus, links contained on such a page remain valid. If a user clicks on such a link, the corresponding page will be displayed in the same way. The links on the left hand side will remain unchanged. Thus the hyperbook's right frame behaves like a normal web-browser whenever external documents are presented to the user. For the Sun tutorial page “Declaring Member Variables” in Figure 4, a link to an example as well as to a course unit is generated.

Modeling the index

To relate the different types of text units, to support student's goals, or to provide guidance, thus for enabling the different adaptation features of the KBS hyperbook system, we have introduced an indexing concept, which will be explained in detail in a later section. In the conceptual model, we see the index concepts, which are called *knowledge items* (KIs), and their relations to the glossary, areas, portfolios, goals, and text units (see Figure 1). Each of these concepts is indexed with a set of such knowledge items.

Integrating Student Projects: portfolios

In order to support project-based learning as described for example in [Henze et al., 1999], our conceptual model contains the concept *portfolio* (see Figure 1). As discussed in [Duschl and Gitomer, 1991], assessment based on portfolios realizes the idea that project results can be used to represent and to assess what topics / concepts a student has successfully applied or learned.

We model a portfolio by relations between project results (portfolio elements) and those topics / knowledge items which have been used for these parts. In our conceptual model, this is expressed by a relationship between portfolio elements and knowledge items (see Figure 1).

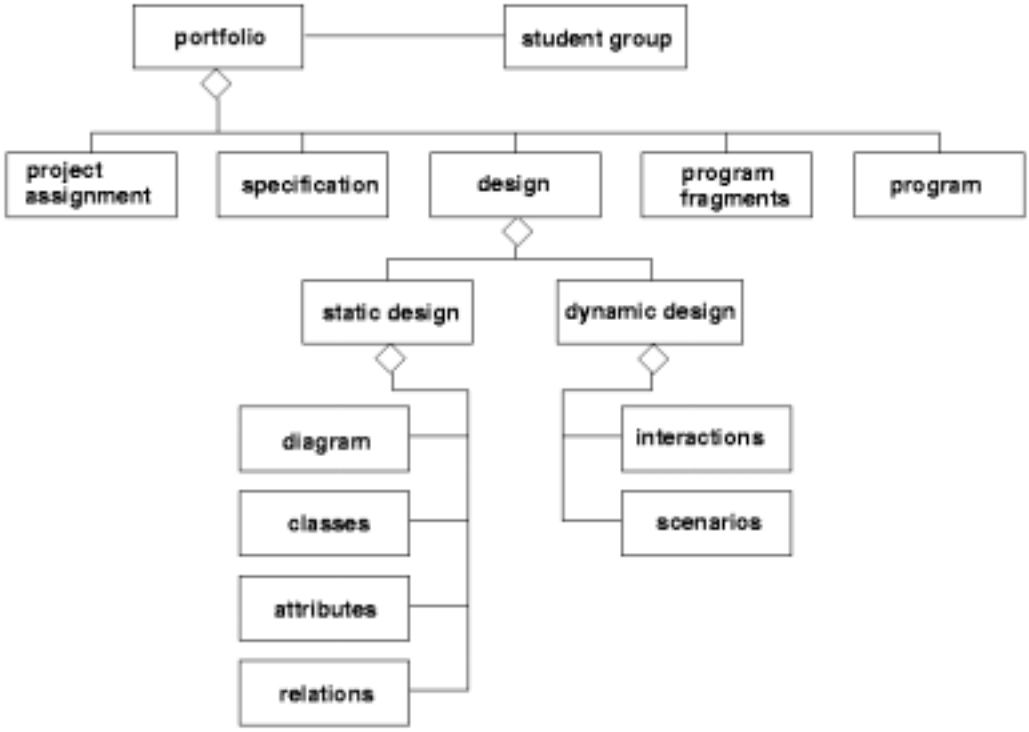

Figure 5: Schematic view of the portfolio concepts

The portfolio elements represent the different parts of a student project. In Figure 5 we see the different kinds of portfolio elements for our CS1 course. This hierarchy mirrors the simplified software modeling process we use in our CS1 course. Important portfolio elements are the specification written by the students, an object oriented design proposal consisting of several subdocuments, documentation of the implementation, and the program code itself. The program code is broken down into different program fragments, each showing the use of some specific concept of the application domain.

To help the students define their own portfolios, we propose a list of knowledge items for the domain of Java as a programming language, plus some knowledge items related to software engineering. There are some KIs which are mandatory for our student portfolios (for example the scenarios and interactions from the dynamic design phase of a project), and a lot of other optional KIs. The students can use a subset of these optional KIs for representing their work in the portfolio individually. A subset of KIs contained in the portfolio can be seen in the following list. KIs marked with an asterisk must be referenced in a student's portfolio, other KIs are optional. As an area consists of several KIs, we are able to easily integrate the portfolios and portfolio elements into the hyperbook on the basis of these KIs. Thus, we define both the basic structure of student projects as well as their connection to the remainder of the course material.

```
object oriented design
 *specification
  static design
    *classes
    *attributes
    *relations
    ...
```

```
user interface
  event model
    *event source
    *event listener
  adapter
  *events
    action event
    text event
     ...
```

An important part of a portfolio is also the presentation of the students who have worked on the project. Therefore the student's homepages are also integrated in the hyperbook (see relation between the concepts *student group* and *portfolio* in Figure 1).

System Architecture

The KBS hyperbook system is implemented entirely in Java. A servlet residing in the Java Web Server (see Figure 6) represents the whole system. The student browses the hyperbook with any HTML browser capable of handling frames, while all necessary processing is done on the server side. Some of the presentation functionalities, such as trails, are also realized by Java client applets.

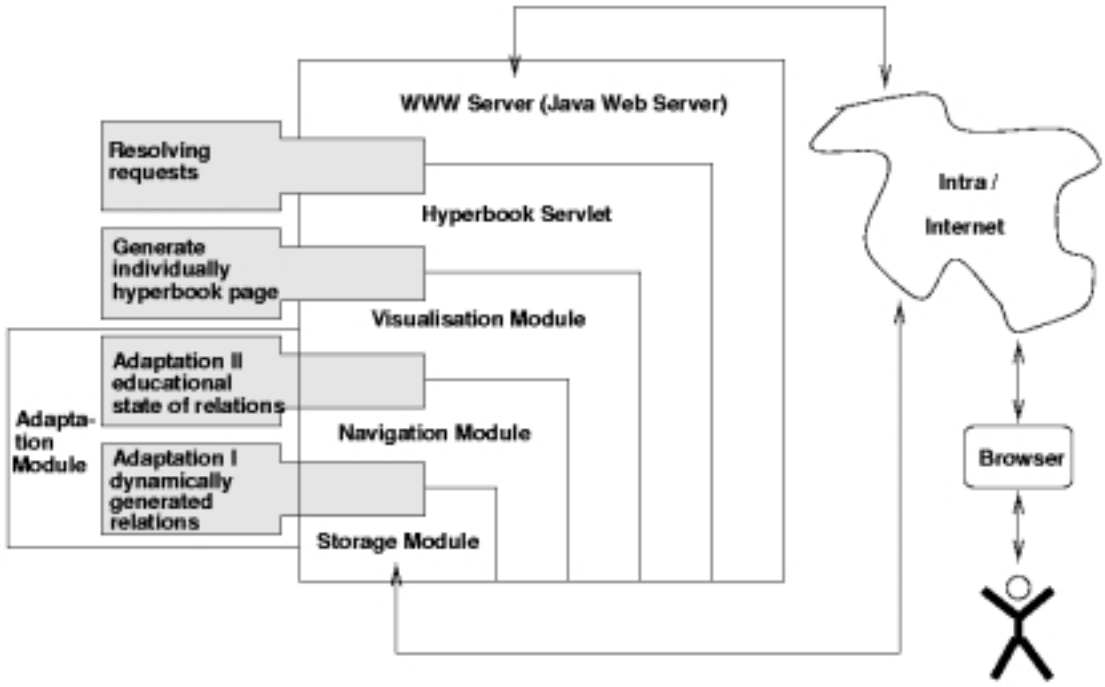

Figure 6: Schematic view of the implementation of the hyperbook system

The user navigates the hyperbook by activating links. These links, however, do not represent static HTML pages. Whenever a hyperlink is activated the name of the corresponding domain object plus the name of the user are passed to the Java servlet residing on the server. This page composition program queries the data base for the URL of the page representing the domain object and for the domain object's navigational possibilities. From this information it constructs a user specific page, and displays it in the user's WWW browser.

On the server side, different components resolve the users' requests. In the following, we will describe the functionality of these components.

Storage Module: The basis of the KBS hyperbook system is the *storage module*. It contains the data of the specific hyperbook, e.g. the instances of the concepts from the conceptual model of the hyperbook in question and implements a query interface. Thus it serves as a data repository and is called by the other modules.

Navigation Module: The *navigation module* describes the navigational possibilities of the hyperbook. It uses the relations defined in the conceptual model to generate navigational constructs for reading, viewing and navigating the resulting hyperbook. All relations in the conceptual model correspond to navigational constructs, which are displayed in a hyperbook

page as hypertext links. Thus, modifications and extensions in the conceptual model are automatically translated to the navigational structure of the hyperbook.

Adaptation module: The navigation module calls the *adaptation module* for adapting the hyperbook to a particular user. The adaptation module dynamically generates relations (dotted lines in Figure 1), trails, next learning steps, and annotates the link structure with the traffic light metaphor.

The adaptation module asks the storage module for the type of the actual concept. If a *text unit* should be adapted, dynamically generated relations to examples and other information, e.g. to the Sun Java tutorial or to the hyperbook library, are returned. For the adaptation of instances of the course concept, a next reasonable learning goal according to the user's knowledge is generated, and the navigation module receives a corresponding relation. If a student defines a learning goal for himself or requests a next learning goal from the hyperbook system, text units containing useful information for this particular project are selected. In addition, a learning sequence containing information necessary for reaching the learning goal is generated, and projects are selected from the project library. The functionality of the adaptation module will be described in detail in the later section about “enabled adaptation”. After having collected both static and dynamic relations, the navigation module asks the adaptation module for the educational state of all the relations of the particular page.

Visualization Module: The *visualization module* then takes the information about ***all*** relations, static and dynamic relations plus their actual educational state, for displaying them as annotated hypertext links on the left hand side of a hyperbook, while the Web page itself is displayed on the right hand side of the hyperbook. There is only one exception: access to the next reasonable hyperbook page, which guides a user to the whole hyperbook, is not displayed as a link in the left frame but is symbolized by the dog's ear on bottom of the right hand side. The arrow on the bottom of the left hand side brings the user back to the current course he is working on.

ADAPTATION COMPONENT OF THE KBS HYPERBOOK SYSTEM

For our KBS hyperbook system we follow a constructivist pedagogic approach, building on project-based learning, group work, and discussions. Such a project-based learning environment leads to particular requirements for adaptation, in order to adapt the project resources presented in a set of hypermedia documents to the student's goals (for a specific project) and to the student's knowledge. It has to support the student learner by implementing the following adaptation functionality:

- Adaptive Information Resources: give the students *appropriate information* while performing their projects, by annotating necessary project resources depending on the student's current knowledge.
- Adaptive Navigational Structure: annotate the navigational structure in order to give the student additional information about appropriate material to explore or to learn next.
- Adaptive Trail Generation: provide guidance by *generating* a sequential trail through some part of the hyperbook, depending on the student's goals.
- Adaptive Project Selection: provide *suitable* projects depending on the student's goals and knowledge.
- Adaptive Goal Selection: propose *suitable* learning goals depending on the particular student's knowledge.

The student modeling component of KBS hyperbooks has to fulfill various tasks. On the one hand it has to enable the above stated adaptation functionality. On the other hand, it has to

enable further adaptation functionality which depends on the openness of the KBS hyperbook approach: Information resources located *anywhere* in the WWW should be included in the curriculum of the student's work with the hyperbook, explanations and examples can originate from the hyperbook's libraries or from any other location in the WWW.

Modeling the Knowledge Domain

The connection between the KBS hyperbook system and the user modeling component is based on indexing any kind of information resources (HTML pages of the hyperbook, projects, examples, web pages, etc.) in the hyperbook. The index concepts are called *knowledge items* (KI). Knowledge items are similar to the domain model concepts used in [Brusilovsky and Schwarz, 1997] or the knowledge units in [Desmarais and Maluf, 1996].

Example 1: Knowledge items (KI) in the CS1 hyperbook are, for example, `if`, `while`, `classes`, `datagram_socket`, `run_method`, etc. A KI may also be compound like `control_structures`, which is a supertopic of `while`.

In order to model the learning dependencies of the knowledge items we put a partial order on the set of all KIs, where $KI_1 < KI_2$ denotes the fact that KI_1 has to be learned before KI_2, because understanding KI_1 is a prerequisite for understanding KI_2. For example, to understand the KI `control_structures`, it is necessary to know about the KIs `branching` and `looping`, thus

looping < control_structures and branching < control_structures.

To identify relevant knowledge items of the application domain of a hyperbook, we first specify the main knowledge concepts of the book and refine these knowledge concepts until we reach the step where further decomposition leads to knowledge chunks which cannot be explained and understood without referring to an embedding context. The finding of knowledge items is a heuristic which must be done by one or several of the authors or by an domain expert. For example, the main knowledge items from the CS1 hyperbook course correspond to the ACM Computing Classification System (1998 version[1]).

Modeling the User's Knowledge

The knowledge of a user is modeled as a *knowledge vector* (KV). Each component of the vector is a conditional probability, describing the system's estimation that a user U has knowledge about a topic KI, on the basis of all observations E the system has about U:

Definition 1 (Knowledge Vector) KV:

$$KV(U) = (P(KI_1|E), P(KI_2|E), \ldots, P(KI_n|E)),$$

where $KI_1, \ldots, KI_n$ are the knowledge items of the application domain and **E** denotes the evidence the system monitors about U's work with the hyperbook.

Observations about the student's work with the hyperbook are stored for each KI. Each observation expresses the grade of knowledge the user has on a KI. We use four grades:

[1] http://www.acm.org/class/1998/css98.html

A student can have

- "expert's knowledge" on a KI; rating **E**: excellent
- "advanced knowledge"; rating **F**: with some difficulities but mainly excellent
- "beginners knowledge"; rating **A**: with many difficulties – seems not to master the concept, or
- "novice's knowledge"; rating **N**: not ready for this concept yet.

Thus, the KIs are, on the one hand, concepts describing the application domain of a book, on the other hand , they are random variables with the four discrete values **E**, **F**, **A** and **N**, coding knowledge grades. The evidence we obtain about the student's work with the hyperbook changes with time. Normally, the student's knowledge increases while working with the hyperbook, although lack of knowledge is equally taken as evidence. Since every kind of observation about a student is collected as evidence, the knowledge vector gives - at each time - a snapshot of the student's current knowledge.

Bayesian Network Engine: Calculating Probabilities

So far, we have seen how to use knowledge items for describing the knowledge domain, and for describing a student's knowledge.

Using probabilities for giving an estimation about a student's knowledge is a very intuitive approach. We give an estimation about the student's knowledge on topic *X* by calculating the conditional probability that *X* is known to this student under the condition "evidence", where evidence is the previously detected information about this student.

As we already used conditional probabilities for describing the student's knowledge, it is obvious to look for inferring mechanisms which allow us to handle networks with dependent random variables. *Bayesian networks* (BN) are useful tools for inferring in graphs with dependent vertices. We use such a BN to calculate a probability distribution for each KI, thus for calculating the knowledge vector of our users.

Definition 2 (Bayesian network (BN)) A Bayesian network is a directed, acyclic graph with the following properties:

- Each vertex in the graph represents a random variable.
- There is an edge from X to $Y \neq X$, whenever Y is dependent on X.
- Each vertex is labeled with a conditional probability table (CPT) that quantifies the effect of its parents. The outneighbours of some vertex are called **children**, the inneighbours are called **parents**. A vertex without outneighbours is called **root**.

To construct a Bayesian network which calculates the probability distribution for each KI of a hyperbook for a particular user, there are two main steps to take: First, generating some acyclic graph which contains the knowledge items as vertices and the learning dependencies between them as edges. Second, defining probability tables for all vertices. For details on the design and implementation of the Bayesian network engine underlying the adaptation component please refer to [Henze and Nejdl, 1999b, Henze, 2000].

Interpreting the Conclusions of the Bayesian Network

After having found an inferring mechanism for the knowledge model, we have to interpret the results of the calculation: how can we classify a user's knowledge as "expert" or as "beginner"? Compared to a continuous variable, we look for the maximum of the distribution, see Figure 7. A knowledge item K is for example "**well_known**" to a user if :

$$P(K=E) + P(K=F) \geq P(K=A) + P(K=N)$$

and it is “excellently_known” by the user, if :

$$P(K=E) \geq P(K=F) + P(K=A) + P(K=N)$$

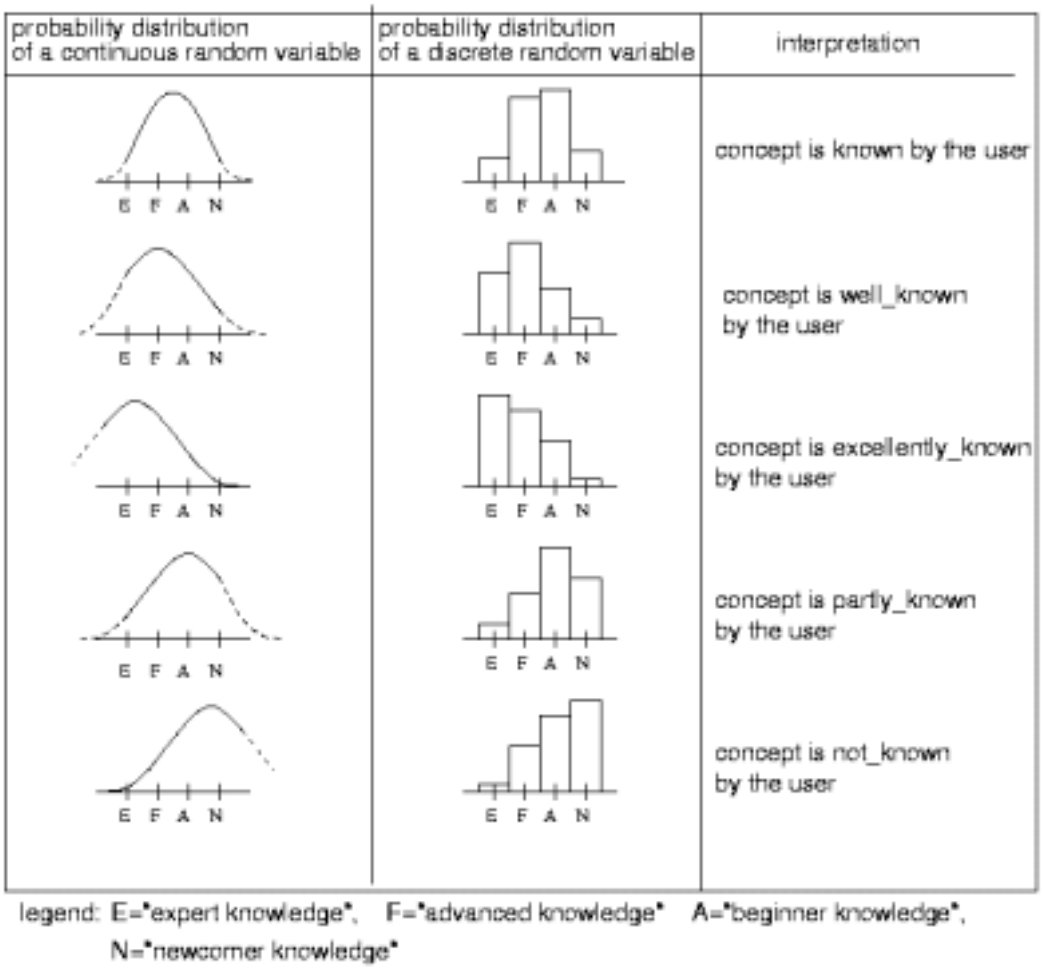

Figure 7: Interpretation of the probability distributions given by the Bayesian network

The Bayesian Network of the CS1 Hyperbook

A part of the Bayesian network of the CS1 hyperbook can be seen in Figure 8. This Bayesian network contains about 291 nodes. The Figure shows the system's estimation of the knowledge of a new user who has no a priori knowledge about the topics concerned with Java. Probabilities assigned to the vertices `algorithms`, `searching`, etc. can also be seen in this snapshot.

Discussion: Bayesian Inference in User Modeling

Bayesian networks are very useful in user modeling since they enable us to manage uncertainty in our observations and their conclusions. For making observations, we currently use four values but more values and thus finer grades for distinguishing knowledge are possible as well. For making conclusions, we estimate the conditional probability that a student knowing a KI_1 with grade expert will know a KI_2 with grade expert, advanced, beginner or novice. The Bayesian community states that conditional probabilities are relatively easy to estimate [Pearl, 1990].

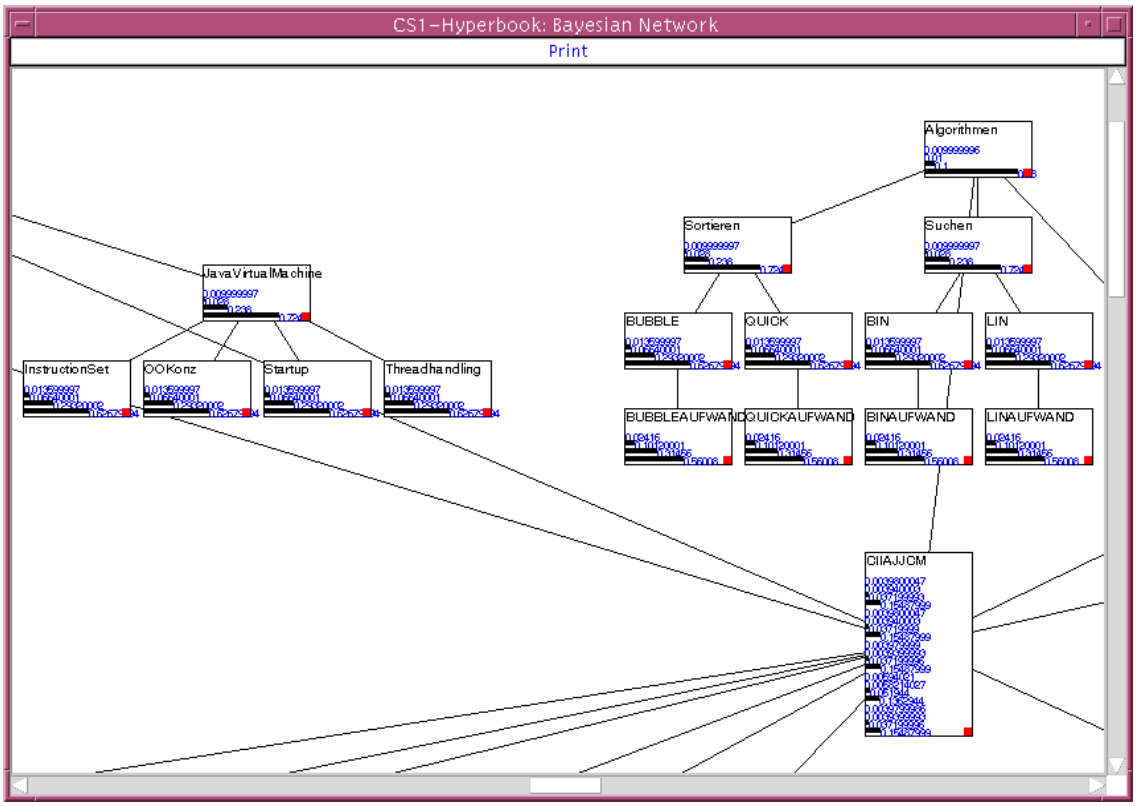

Figure 8: Part of the Bayesian network for a CS1 hyperbook

Another advantage of using a BN is that we can make inferences on top of the complete knowledge model of the application domain. This knowledge model contains all necessary prerequisites for a particular knowledge item, models dependencies among knowledge items, and is able to infer, for example, that prerequisite knowledge of a KI has already been acquired by a user if the KI itself is understood by the user.

By using a BN, it is possible to use observations about the user's work with the hyperbook and with the hyperbook projects to update the system's estimate of the user's knowledge in the way our application requires it. For example, if the system's estimate of the user's knowledge is too pessimistic, and the user solves an advanced project which the hyperbook had thought to be too difficult for him, the system can use this observation to update its estimate, based on the successful completion of the project unit and the indexing of project units by knowledge items. On the other hand, if we observe an advanced user failing to understand some simple concepts, then the BN can selectively change its estimate of this user with respect to these concepts, without classifying him as a complete beginner, and can suggest specific project units for learning these concepts.

Another advantage of using BNs is the managing of uncertainty in our observations and their conclusions: We can express *the belief* that knowledge about topic X requires knowledge about topic Y in the conditional probability tables in the BN.

BRIDGING THE GAP BETWEEN LEARNING MATERIALS AND STUDENT MODEL: INDEXING WITH KNOWLEDGE ITEMS

Each information resource is indexed by some set of knowledge items describing the content of the resource. These resources can be some HTML pages, examples, projects, etc. The origin of an information resource is not relevant for indexing, only the content defines the index.

Definition 3 (Content Map) Let $S \neq \emptyset$ be the set of all KIs, and let H be a set of HTML pages. Then

$$I : H \rightarrow P(S) \setminus \{\emptyset\}$$

is the content map, which gives for each information resource in H the index of this resource, e.g. the set of KIs describing its content (P(S) is the power set of S, exluding the empty set guarantees that each page is described by at least one KI.).

To identify the index of an information resource we plan to scan the text for keywords or phrases. Actually, the indexing is done by the author of an information resource by hand.

An HTML page is indexed by those KIs which describe the content of the page. There is no difference in indexing information resources belonging to the hyperbook itself and in indexing information resources belonging to some server anywhere in the web. Only the content of this HTML page is relevant for indexing.

Indexing introductory pages

In general, each KI of the knowledge model can be used for indexing. There are some introductory concepts in the set of knowledge items, each beginning with the phrase `what_is*` (or `what_are*`). These concepts are useful for describing entry points to more detailed information. Authors often prefer to give a short overview about a topic before going into a detailed description. Furthermore, this new convention enables the system to generate a glossary.

Example 2 To explain applet programming in Java, an author starts with a description of stand-alone programs which can be distributed over the WWW. This information resource H does not contain information about actually writing Java applets, but useful information for understanding the idea of applets. The index of this page contains only one KI: The `what_is`-KI of the KI `java_applets`. For example, in the CS1 hyperbook, I(H)={`what_are_applets`} for this resource H.

Indexing detailed information

If an information resource contains a detailed description of a topic, the corresponding KI from the knowledge model is used for indexing.

Example 3 An information resource H describing the use of the main method of some Java application will be indexed by the KI `public_static_void_main`. If this page also contains information about reading command line parameters, the index will contain the KIs `public_static_void_main` and `input_parameter`.

The dependencies of the knowledge items can be used for abbreviating the indexing process by using higher level concepts instead of many lower level concepts.

Indexing Portfolios

A portfolio concept is indexed by the KI which describe the particular content of this portfolio page.

Indexing Examples and Projects

The indexing of examples and projects is similar to indexing arbitrary hyperbook pages: Each topic explained in the example is indexed by the corresponding KIs, higher level KIs can be used as well. To emphasize specific topics as *very important* or *very well explained* in an example, the author of a project can provide weights with each KI indexing the project. Such a weight is the percentage of a KI in this particular project.

Example 4 The "threads in applets" project shows the use of threads for loading data in applets. It is indexed by the KIs `java_applet` (30 %) and `single_thread` (70 %).

Update: Observations about Users

Several systems detect the fact that the student reads some information to update the estimate of his knowledge (e.g. [Brusilovsky and Schwarz, 1997]). Some of them also include reading time or the sequence of read pages to enhance this estimation. While this is a viable approach, it has the disadvantage that it is difficult to measure the knowledge a student gains by "reading" an HTML page [Brusilovsky, 1996]. In the current state of our development, we decided to take into account neither the information about visited pages nor the student's path through the hypertext. Instead we use only the projects for updating the system. This is motivated by the teaching approach for our hyperbooks. For motivating the students to explore the learning material by doing some projects, the hyperbook is only updated when we have some results about a student's performance in a project.

The updating can happen in two ways: Either we ask the student for direct feedback after working on a project. The student then judges his own performance by selecting one of the categories "topic was easy - I mastered it effortless", "topic was okay - but some problems were arising", "topic was hard - I had a few ideas but could not get the thing right" and "no idea about this topic at all". These four categories correspond to the grades of knowledge a user has on a KI. The second way is to ask some experts for judging the student's project performance. We use the latter in our CS1 course: here the mentors which coach the student's project work give the judgment.

Indexing Constraints

For obtaining a correct indexing of the hyperbook, the index set has to obey the following constraint, which is motivated by the project-based learning approach underlying our hyperbooks. As we have seen in the last section, we only use the performance of students in projects as indicators for their learning progresses. Thus, projects are the only source for updating the hyperbook. This requires that each KI must be contained in an index set of some project:

$$\forall KI : \exists \text{ project } P \text{ with } KI \in I(P).$$

Discussion

We have seen how to use knowledge items for indexing all kinds of information, belonging to the hyperbook, or located anywhere in the WWW. The use of an indexing concept in student and user modeling in this way is different than other indexing concepts in student and user

modeling [Brusilovsky, 2000]. In most other adaptive hypermedia systems, dependencies like prerequisites or outcomes are directly connected to the information resources themselves.

We separate knowledge and information, as we model learning dependencies solely on the set of KIs of a hyperbook. The connection between the student modeling component and the hyperbook system is the content map (see definition 3), which maps each information resource to a set of KIs.

This separation is advantageous in many aspects. As the KBS hyperbook system allows different authors to write parts of a book, they become independent from the work of others: They can write (and index) their information entries without caring about the other content of the hyperbook. The KBS hyperbook system is an open hypermedia system, allowing use of information resources located anywhere in the WWW. As all information resources are equal in the sense that they only need to be indexed for being integrated in a particular hyperbook, the system's openness is enabled by the indexing concept, too. In addition, as we will see in the next section, all kinds of information resources from arbitrary origins are fully integrated and adapted to the student's needs: We can propose programming examples in the WWW, generate reading sequences which contain material of the hyperbook library and the WWW, calculate the educational state of HTML pages in the WWW according to the student's actual knowledge state, etc.

Clearly, the use of a separate knowledge model makes the hyperbook system robust against changes. If we add additional information pages or change contents, we only have to (re-)index these pages accordingly. No further work has to be spent on updating other material, as would be necessary if knowledge, and thus reading or learning dependencies, were coded in the material itself. This implementation enables us to apply different inference mechanisms to the student modeling component. The inference technique we currently use for the KBS hyperbook systems is Bayesian inference.

ENABLED ADAPTATION

In this section, we describe the facilities of the adaptation component, which are enabled by the indexing and knowledge estimation approach proposed in the last two sections.

Link Annotation

Annotation of links is very useful if a user wants to browse through the hyperbook. Links can be enriched by additional information: a heading, a short abstract of the concept it links to and a hint indicating the educational state of this link. Heading and short abstract of a page belong to the meta description of an information resource and are displayed by the KBS hyperbook system whenever a link is generated. For calculating the relevance of information for a student according to his actual knowledge state, we use a simple *traffic light metaphor* for annotation: Links are marked as ready_for_reading (green ball in front of a link), not ready_for_reading (red ball) or already_known (grey ball) to help the user select appropriate information units.

Calculating the Educational State of a Link

The ready_for_reading-function calculates for different types of information resources the educational state of this information for a specific user and his current knowledge. The following definitions are used by the ready_for_reading function:

A KI is **child_known**, if it is known, well_known, or excellently_known, a KI is **parent_known**, if it is well_known or excellently_known. Recall that we obtain the estimations

known, well_known, etc. by interpretating the probability distributions of the knowledge items in the Bayesian network.

For calculating the educational state of a HTML page H, we look at the index of this page. A page is recommended for reading, if *all* child concepts of *all* KIs of the index are "child_known" which means that all prerequisites required to understand the page H are at least known to the user.

H is **ready_for_reading** for a student, if

$\forall$ K $\in$ I(H) : ($\forall$ C, C child of K: C is child_known)

H is **already_known** to a student, if

$\forall$ K $\in$ I(H) : ($\forall$ P, P parent of K: P is parent_known)

If H is neither ready_for_reading nor already_known, H is not recommended for reading yet. These functions are the same for either information resource: Glossary concepts, example pages, HTML pages in the WWW, HTML pages of the hyperbook itself, etc.

Access to Relevant Information: Trails and Information Index

A student often needs information about specific topics, but lacks prerequisite knowledge for understanding them. For example, a student wants to work on a project about `algorithms` but does not understand `control_structures` or `methods`; in this case it would not help to start reading the information unit about `algorithms`. To support the student, the system compares his actual knowledge with the knowledge required to understand the topic in question. If the student lacks some prerequisites, the system generates a sequence of information units - a trail - that guides his learning towards the selected topic.

Generating a Learning Sequence

Generating such a trail is implemented by a depth-first-traversal algorithm which checks the system's estimation of the student's knowledge of those KIs that are prerequisites for the actual goal. The algorithm checks whether all prerequisite knowledge is sufficiently known by the student. If not, the corresponding information units of the hyperbook are marked. Afterwards, a sequence of all marked units is generated which guides the student consistently through the topics up to the selected ones.

Learning Sequences are generated, if a user

- selects a goal, or
- enters a project or example.

Figure 10 shows the project "synchronizing threads" which is presented to a beginner. The trail leading to the required information is therefore very detailed.

Generating a Glossary

Each hyperbook has a glossary which is generated from the knowledge model. A KI is used as a glossary item, if this KI is a `what_is*-` KI or if it is a leaf node in the Bayesian network.

Generating an Information Index

The hyperbook also provides direct access to information needed for the actual task (information goal or project) of a student. Relevant information is selected on the basis of the index concepts of the actual task. Access to the information is given by a sorted index. Each link in this index is annotated according to the student's knowledge, using the traffic light metaphor.

A list of related or relevant information is generated if the student

- reads a glossary item: propose relevant examples, information of the Sun Java tutorial and information in the hyperbook;
- reads a page of the hyperbook: propose relevant examples and links to the Sun Java tutorial;
- reads a Sun Java tutorial page: if we display a resource from the WWW, we provide links to relevant examples and to the corresponding information in the hyperbook itself;
- studies examples: generate access to relevant hyperbook pages and Sun Java tutorial pages.

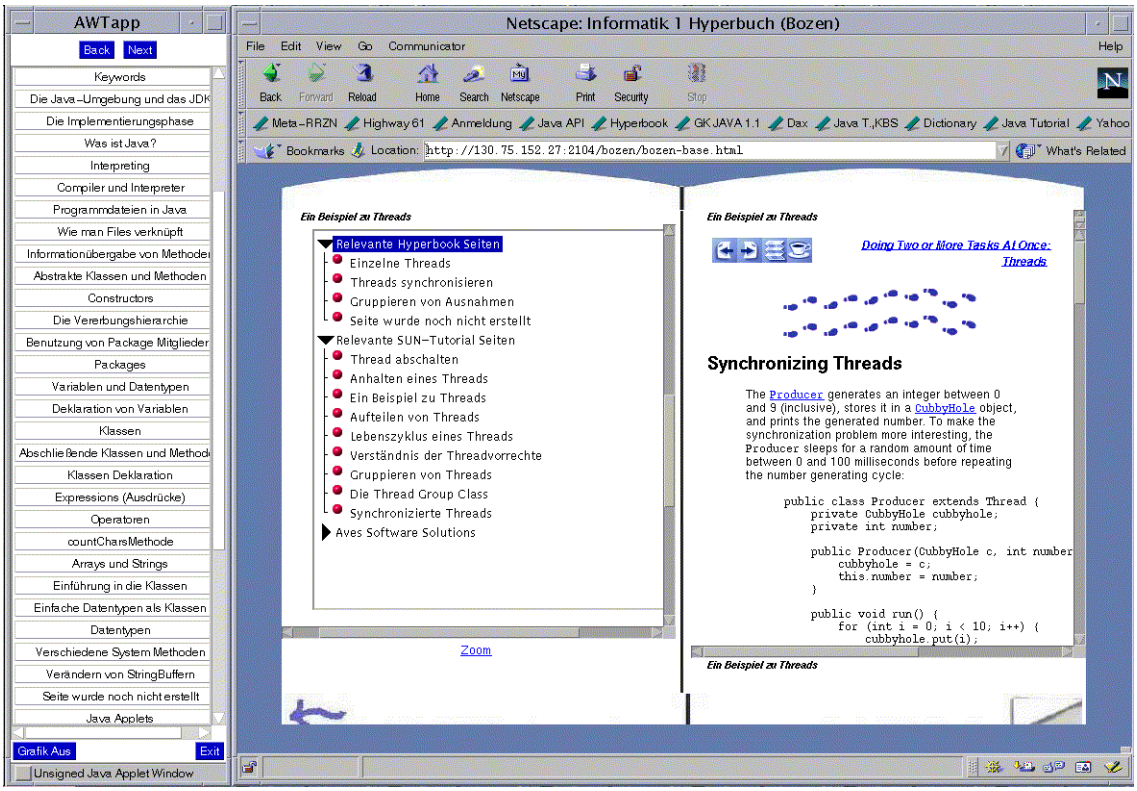

Figure 10: The project "synchronizing threads" is presented to a beginner.

In Figure 10, we can see the information index belonging to project "synchronizing threads". The index contains the knowledge which is used in the project. Prerequisite knowledge required for working on the example is contained in the accompanying trail.

Direct Guidance

If a student wants more guidance during the learning with the hyperbook he may ask the hyperbook for the next reasonable learning step. This request is answered by determining a suitable learning goal, depending on his current knowledge. Based on this goal, the hyperbook can propose a suitable project, a set of HTML pages with relevant information, and a trail leading to that goal. A goal is defined as a set of knowledge items. To determine the next suitable learning goal, a sequential trail covering the whole hyperbook is calculated. For each

item of this trail the system's estimation of the student's knowledge is checked. If the student fails to know some knowledge item then it is proposed as the subsequent suitable goal.

Guide a student's learning process by

- proposing reasonable learning goals, including links to suitable projects and a learning sequence;
- proposing the next page to read; and
- generating reading sequences for projects or user defined goals.

Goal Based Learning

Sometimes, a reader of a hyperbook has a certain goal. Maybe he requires specific information, searches for examples illustrating a certain topic, or wants to learn a special part of the hyperbook. He can inform the hyperbook directly about his current goals by selecting some of the knowledge items out of a list. The hyperbook takes these user defined goals and selects examples, which explain the concepts of this goal, gives access to relevant information, and generates a reading sequence containing the information to reach the goal.

Project Based Learning

In order to select suitable projects for a student, the hyperbook contains a project library. Each project is indexed by the KIs that have to be understood in order to successfully accomplish the project. Since we use a Bayesian network for modeling of the student's knowledge [Henze and Nejdl, 1999a], we do not have to include prerequisite knowledge items, because they are already taken care of by the dependency structure modeled in the BN.

A project is useful for a student in his current knowledge state and his situation, if

- The KIs comprising the student's goal are sufficiently contained in this project, and
- All KIs which are not pat of the student's goal but necessary for the project, are understood well enough.

These requirements determine the selection criteria for finding an appropriate project for a student that simultaneously helps the student to achieve his learning goal and reflects his current knowledge state. They are implemented by two algorithms [Henze and Nejdl, 1999a]: The first one calculates how well a project *matches* the goal of a user (*project-goal-distance*). The second one determines whether the actual knowledge of a user is sufficient for performing the suggested project without too many difficulties (*fitness*). For example, a student who is interested in learning simple control structures in Java will have difficulties with a project that uses control structures to build a graphical user interface provided that he has only beginner's knowledge about graphical user interfaces.

The hyperbook selects the projects by comparing the weighted sums of these two measures. The weights allow us to emphasize either one of the aspects *matching* and *fitness*.

Matching: How well does a project fit to a student's goal?

We implemented a distance function that calculates the project-goal distance between a project P and the actual goal G, based on the KIs contained in the goal and their relevance in the project.

Each KI contained in the goal is assumed to have a relevance of 100. The relevance of a KI for a project is defined by its percentage in relation to the whole project. A short distance means

that this KI is very important for performing the project, while a large value represents the fact that the KI is not very relevant for the project. For every KI of the goal G that is not contained in the project, this distance is set to a maximum value of 100. Thus, for all $KI \in G$, we have

$$\text{distance}(KI, P) = \begin{cases} |100 - \text{relevance}(KI, P)|, & \text{if } KI \in P \\ 100, & \text{other cases} \end{cases}$$

where relevance (KI, P) is a percental distribution on the index of the project which has to be defined by the project's author. The project-goal-distance for a project and a given goal is then calculated as the mean value of all these distances:

$$\text{project-goal-distance}(G, P) = \frac{\sum_{KI \in G} \text{distance}(KI, P)}{|G|}.$$

Fitness: How about those parts of a selected project that do not belong to the student's current goal?

The second algorithm determines the *fitness* of a user U for a project. To determine this fitness we evaluate the knowledge of the user concerning those parts of the project that do *not* belong to the user's goal G. This enables us to select projects that are based on prerequisites already known by the user, and thus lead him as fast as possible to his goal.

$$\text{fitness}(G, P, U) = \frac{\sum_{KI \in I(P) \setminus G} \text{knowledge}(KI, U)}{|I(P) \setminus G|}.$$

Recall that I(P) denotes the index of the project. The function knowledge(KI, U) is the system's estimation of the user's knowledge with respect to this KI:

$$\text{knowledge}(KI, U) = (P(KI=E) \cdot 1 + P(KI=F) \cdot \square + P(KI=A) \cdot \square) \cdot 100.$$

Integrating Portfolios

If we observe that a student has studied some portfolio, we integrate links to parts of this specific portfolio whenever it is useful. For example, if the student reads in the glossary he finds links to the relevant part of this particular this portfolio. The assumption is that it is advantageous for a learner to see the “use” of a concept in an already known scenario: in this case the portfolio with its problem descriptions and solutions. For example, a student has studied the portfolio of the student group “Duck”. Then links to relevant parts of the portfolio are generated and presented to the student.

RELATED WORK IN ADAPTIVE EDUCATIONAL HYPERMEDIA

In this section we will compare the KBS hyperbook system to other approaches in educational hypermedia which provide adaptational functionality similar to those in hyperbooks. A review on adaptive educational hypermedia systems [Henze, 2000] showed that especially the systems ELM-ART [Brusilovsky et al., 1996a, Weber and Specht, 1997], INTERBOOK [Brusilovsky et al., 1996b, Brusilovsky and Schwarz, 1997], PT [Kay and Kummerfeld, 1994, Kay and

Kummerfeld, 1997], and AHA [de Bra, 1996, Calvi and de Bra, 1997], focus on similar aspects. In the following we will shortly describe those four systems and compare them to KBS hyperbook in the dimensions of user characteristics and of link level adaptation.

The ELM-ART system [Brusilovsky et al., 1996a] and its successors ELM-ART II [Weber and Specht, 1997] and INTERBOOK [Brusilovsky et al., 1996b] are some of the first adaptive hypermedia systems which were used in the WWW.

ELM-ART uses an episodic learner model [Weber and Möllenberg, 1995] (ELM) for diagnosing complete and incomplete problem solutions. Concepts in ELM-ART are related to each other by their prerequisites and outcomes. Thus, a conceptual network is constructed. Observations about the user are made by monitoring the visited pages: the concept corresponding to a visited page is marked in the conceptual network as known. For annotating the links, the authors use the traffic light metaphor. A red ball indicates pages which contain information for which the user lacks some knowledge, a green ball indicates suggested links, etc. ELM-ART also contains interactive examples, which can be translated with a Lisp compiler via the web.

ELM-ART II was developed for translating normal textbooks into electronic textbooks. ELM-ART II improves the knowledge representation of ELM-ART. The conceptual network is hierarchically organized into lessons, sections, subsections and terminal pages. Each unit in the conceptual network has a slot with the text for the page and the information for relating this page to other units, information required for interactive tests, and for programming problems. The individual learner model stores visited pages, solved tests, and solved programming problems by marking the corresponding concepts in the conceptual model as “known”. Direct guidance is provided by a “next best” button, help is proposed by finding the most relevant example from the individual learning history, based on a diagnosis of the programming code of the learners solutions.

The systems are able to make inferences about the users knowledge based on the marked concepts in the conceptual model: All prerequisites of the known concepts are also marked recursively as known.

In the INTERBOOK project [Brusilovsky et al., 1996b], electronic textbooks are created on the basis of a hierarchically structured MS-Word file. Several steps, such as creating a list of domain concepts, structuring and annotating the pages with outcome and prerequisite knowledge, translation to HTML and parsing the information into a Lisp structure have to be done to obtain an INTERBOOK.

INTERBOOK uses both a domain and a user model. The glossary and the textbook are based on the domain model. Each textbook unit is indexed with some domain model concepts. These concepts have different roles. Some of the concepts describe the *outcome* knowledge which a user has after reading the page, and others the prerequisite or *income* knowledge which is necessary to read the page.

INTERBOOK supports adaptive annotation of links by using the traffic light metaphor. It implements a prerequisite based help by presenting an annotated list of pages that contain prerequisite information. Page sequencing is done in three steps. First, the system computes overall scores for the supposed state of knowledge for each concept. Based on these scores the system decides whether a concept is already well-learned or not. Second, the system decides which pages contain suggested teaching operations or which have missing prerequisites. Links to concepts and sections of different educational states are annotated by different icons. Third, the system selects the most optimal page among all available pages that introduce unknown concepts and that are missing no prerequisites. To all pages, a certain priority for presentation is assigned, based on a default value and modified according to the state of knowledge of the required and introduced concepts.

An approach to implement a user adaptive interface for the INTERBOOK project is described in [Brusilovsky and Schwarz, 1997]. A domain model and a student model are used for representation. Each interface feature is treated as a domain concept, and each hint as a

learning unit. By applying a task sequencing algorithm similar to the one mentioned above, a sequence of interface features to be learned and a sequence of hints which will be presented to the user are generated.

PT (personalized text system) is a textbook for learning the programming language C [Kay and Kummerfeld, 1994,Kay and Kummerfeld, 1997]. It uses a conventional book about C as a base for generating the hypermedia system. The course which was supported with PT is a C programming course for Pascal programmers.

PT uses a stereotypical user model of the target audience (Pascal programmers) together with an individual model. The stereotype provides certain values for the knowledge components which initialize the user model. In the individual model, the knowledge values of the individual user are stored during their work with PT. For enabling adaptation, PT uses similarities between Pascal and C for presenting information to the users. Preprocessor commands are added into a raw HTML page, e.g. `#define PASCAL 3`, `#define active-learner 1`, `#if PASCAL > 2`, etc., from the user model. The if preprocessor commands in the raw HTML page are used to control which parts of the page are passed to the user. The same preprocessor technique is used for the selection of links.

A free web-course about *Hypermedia Structures and Systems* [de Bra, 1996, Calvi and de Bra, 1997] is implemented in the AHA system. AHA (Adaptive Hypermedia Architecture) can be used to generate conditional text, and to adapt the link structure by link removal, link hiding and link annotation. Preprocessor commands in the HTML pages are used by CGI-scripts to filter content fragments of a page and thus enable content adaptation. The same preprocessor technique is used for link adaptation.

characteristics of the user	know-ledge	goals	prefe-rences	back-ground	expe-rience	learning speed
AHA	X					
ELM-ART	X				X	
INTERBOOK	X	X			X	
KBS	X	X				X
PT	X		X	(X)		

Figure 14: Characteristics of a user taken into account by the five hyperbook-like approaches

Figure 14 shows the characteristics of a user which are taken into account by these five hyperbook-like approaches. None of them uses the background as information source, as it is done for example in EPI-UMOD [Rosis et al., 1992] or ANATOM- Tutor [Beaumont, 1994]. PT assumes that all users of the system have the same background. Thus it is a general assumption rather than a background characteristic. In these five hyperbook-like approaches, content-level adaptation is only done in the AHA system. The methods used for navigation support can be seen in Figure 15.

The approaches in INTERBOOK and ELM-ART II are by means of their concept and idea very similar to KBS. The adaptation features, like link adaptation, goals, page sequencing, are present in each of the three systems. However, the implementation strategies to obtain these functionalities vary.

Comparing INTERBOOK with KBS, we see a difference in the domain description. INTERBOOK uses a conceptual network of the domain, while KBS uses both conceptual model and knowledge model. Therefore the introduction of slots as used in INTERBOOK is not necessary in KBS, as assumptions about prerequisite and outcome knowledge are based upon

the knowledge model. The algorithm for page sequencing proposed by INTERBOOK proposes pages if they are recommended to visit; it is not explicitly taken into account if they also fit to the previously read page. The page sequencing algorithm in KBS, which is invoked whenever a student defines or selects a learning goal, generates a sequence of pages that reflects a didactical ordering in order to reach the current learning goal.

navigation support	page se-quencing	ad. nav. support	user ob-servation	goal support	example support	project support	open-ness
AHA		hiding	tests, readpages		section		
ELM-ART	course	annotation	tests, readpages		section		
INTERBOOK	page, course	annotation	tests, readpages	user defined	section		
KBS	page, course	annotation	direct feedback	user def., proposed	page	knowledge, goals	WWW
PT		hiding	tests		section		

Figure 15: Methods for link level adaptation in the five hyperbook-like approaches

The example selection in KBS is done for each page contained in the hypermedia system. The example and project libraries of a hyperbook are checked for examples illustrating the content of the actual page. INTERBOOK or ELM-ART II choose examples based on the chapter-section-subsection hierarchy.

The preprocessor techniques used in PT and AHA differ from the KBS implementation, too. However, navigational help strategies are comparable.

We can see, that the indexing concept underlying the KBS is more general than the techniques in the other hyperbook like approaches, since it separates documents from didactical information or reading orders. This enables the KBS system to integrate information from arbitrary origins, because the documents have not been adjusted to the other, existing documents. In addition, KBS is able to select and propose projects and supports the learner's work with these projects.

CONCLUSION

In the previous sections, we have discussed our current system, which is able to refer to external information resources and can adapt these additional information resources to the user's knowledge, goals and preferences. The system makes no difference between local and distributed materials, its adaptation functionality applies equally well to internal and external data.

The approach we have described is based on describing the content of data by relating it to index entries in a content map. Knowledge or learning dependencies are separated from content and are contained in an extra model. This approach allowed us to integrate e.g. the Sun Java Tutorial into our hyperbook about Java programming, and to integrate student projects into the learning materials. While investigating and implementing our approach, we have come up with some challenges for further research in this area which we discuss in the following.

Learning Dependencies - always the same?

In the hyperbook system, we store learning dependencies in a separate model: the knowledge model. This model contains the prerequisite knowledge required to understand some concept, as well as the resulting knowledge. It does not refer to a certain teaching strategy. Nevertheless, assumptions about the sequencing of the learning material are encoded in this knowledge model. This can be made clear for our course on Java programming. While this area is well structured,

the kind of learning an instructor has in mind while designing his material is implicitly given in the way he defines the knowledge model: If the focus is on object orientation, the author usually starts with explaining the ideas of object orientation without referring to any required (pre-) knowledge. However, if the instructor is coming from a structured programming background, he usually introduces object oriented language constructs after discussing structured programming constructs. In this case, information material describing object oriented programming concepts requires previous knowledge about structured programming.

Thus, even if we separate knowledge dependencies from content, we have to deal with the problem, that knowledge, or, in the educational area, learning dependencies are not the same for every instructor, but still depend on the content used in a specific course. Modeling these dependencies explicitly helps us in comparing and integrating these different teaching strategies.

We are currently replacing the knowledge model by a general domain ontology and investigate how different learning dependencies can be expressed in such a general model.

Generating Reading Sequences

Curriculum sequencing is an important adaptational functionality in hypermedia systems. In the case of our *open* hypermedia system, we sometimes have to deal with too much information: The same knowledge is made available by different resources, in our case on local Hannover KBS pages or on SUN Tutorial pages or on pages belonging to our Bozen course. Take, for example, that we have two HTML pages H_1 and H_2 with $I(H_1) = \{X_1, X_2, X_3\}$ and $I(H_2) = \{X_1, X_2, X_4\}$. (Recall that $I(H)$ denotes the index of resource H.) If the system infers that X_1 should be contained in the reading sequence, which of these HTML pages should be selected, which should be skipped? In such cases, the knowledge dependencies - maybe from different knowledge models - as well as the whole reading sequence has to be considered to make an appropriate choice. In the current state of the KBS Hyperbook system, we do not compute reading sequences with content from different origins. Instead, we are generating separate trails, through the local material itself (Hannover and Bozen pages are largely homogeneous) or through the Sun Tutorial.

Integration of Structured vs. Unstructured Materials

Our open hypermedia system has to integrate both structured and unstructured materials. In this paper, we have proposed a solution for integrating material by taking it as “stand-alone information”. We have not used the context and structure, in which these pages are embedded (e.g. that the SUN Java Tutorial uses trails with a hierarchic structure of information elements). In cases where the information in the SUN Java Tutorial is more detailed than our own local KBS pages, these leads to references to a whole lot of SUN Tutorial pages, without making the hierarchic structure of this additional information apparent.

A possible solution for the integration of such structured material can be preselection. In this way, we can for example filter according to depth of information and reference only pages, which are on the same level. Thus, more detailed pages from the SUN Java Tutorial are available only from a top level page, which is referenced from the appropriate KBS page, and not directly linked to the KBS page itself. Also, the additional hierarchy available can be displayed similar to a table of contents for some area. The advantage in this approach is that the structure of the hypermedia system a page originates from, will be presented to the user by the page itself. Drawbacks are the existence of two different structures (though the second structure can sometimes be seen as a direct extension of the first, less detailed structure).

In the case of integrating project portfolios, what is still missing are explanations of why certain portfolio elements are relevant to a specific topic / knowledge item. When the portfolio is used during an exam, and the student can explain this relationship and why he uses a specific portfolio element to illustrate a certain topic, this is ok. However, if the portfolio elements are used as part of the learning materials without any additional explanations provided, it is

sometimes very difficult to use the referenced program fragment as a stand-alone example for a specific topic.

Reuse and Integration of Indices

As adaptive hypermedia systems are becoming more common, reuse and integration of indices from different courses on the same topic becomes an interesting issue. Currently, two courses developed at CTE (Carnegie Technology Education) cover several overlapping areas with our CS1 course, and they also use indexing similar to our content map (though with additional relations) [Brusilovsky, 2000]. Using a comparison of these content maps and the topics contained within them currently seems to us a good starting point for exploring the issues involved in integrating and reusing content maps from different courses but overlapping areas, and even might lead to standardized content maps for different areas, possibly with different learning dependencies.

Furthermore, as the meta data in our hyperbook system are largely compatible with the RDF standard for Web annotation, it would be interesting to come up with standardized means of reuse and exchange of those content maps, giving distributed adaptive hypermedia systems access to the full content of the Web (at least to those parts which are relevant and also are annotated with appropriate metadata information). The meta data used in the proposed system is contained in the draft standard for Learning Objects Metadata (LOM; IEEE P1484.12, Learning Object Metadata Working Group, http://ltsc.ieee.org/wg12/). We use from category *General* the data element 1.6 *Keywords* which carries the result of the content map – thus the set of knowledge items describing the content of the information resource. The knowledge model is coded by data element 7.1. *Kind* (IsBasedOn / Requires).

REFERENCES

Beaumont, J. (1994). User modelling in the interactive anatomy tutoring system ANATOM-Tutor. User Modeling and User Adapted Interaction, 4(1):21-45.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87-129.

Brusilovsky, P. (2000). Course sequencing for static courses? Applying ITS techniques in large-scale web-based education. In *Proceedings of the fifth International Conference on Intelligent Tutoring Systems ITS 2000*, Montreal, Canada.

Brusilovsky, P. and Schwarz, E. (1997). User as student: Towards an adaptive interface for advanced web-based applications. In *Proceedings of the Sixth International Conference on User Modeling, UM97*, Sardinia, Italy.

Brusilovsky, P., Schwarz, E., and Weber, G. (1996a). ELM-ART: An intelligent tutoring system on world wide web. In Frasson, C., Gauthier, G., and Lesgold, A., editors, *Intelligent Tutoring Systems (Lecture Notes in Computer Science, Vol. 1086)*, pages 261-269, Berlin. Springer.

Brusilovsky, P., Schwarz, E., and Weber, G. (1996b). A tool for developing adaptive electronic textbooks on WWW. In *Proceedings of WebNet'96 - World Conference of the Web Society*, Boston, MA, USA.

Calvi, L. and de Bra, P. (1997). Improving the usability of hypertext courseware through adaptive linking. In *The Eighth ACM International Hypertext Conference*, Southampton, UK.

Campione, M. and Wallrath, K. (1999). *The Java Tutorial.*. Addison Wesley, 2nd edition. http://www.javasoft.com/docs/books/tutorial/index.html.

de Bra, P. (1996). Teaching hypertext and hypermedia through the web. In *Proceedings of WebNet 96 World Conference*, San Francisco, USA.

Desmarais, M. C. and Maluf, A. (1996). User-expertise modeling with empirically derived probabilistic implication networks. *User Modeling and User Adapted Interaction*, 5:283-315.

Duffy, T. and Jonassen, D., editors (1992). *Constructivism and the Technology of Instruction*. Lawrence Erlbaum Associates.

Duschl, R. A. and Gitomer, D. H. (1991). Epistemological perspectives on conceptual change: Implications for educational practice. *Journal of Research in Science Teaching*, 26(9):839-858.

Henze, N. (2000). *Adaptive Hyperbooks: Adaptation for Project-Based Learning Resources*. PhD Thesis, University of Hannover. http://edok01.tib.uni-hannover.de/edoks/e002/ 13646791.pdf .

Henze, N. and Nejdl, W. (1997). A web-based learning environment: Applying constructivist teaching concepts in virtual learning environments. In *IFIP 3.3 and 3.6 Joint Working Conference: The Virtual Campus: Trends for Higher Education and Training*, Madrid.

Henze, N. and Nejdl, W. (1999a). Adaptivity in the KBS hyperbook system. In *2nd Workshop on Adaptive Systems and User Modeling on the WWW*, Toronto, Canada.

Henze, N. and Nejdl, W. (1999b). Bayesian modeling for adaptive hypermedia systems. In *ABIS 99, 7. GI-Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, Magdeburg.

Henze, N., Nejdl, W., and Wolpers, M. (1999). Modeling constructivist teaching functionality and structure in the KBS hyperbook system. In *CSCL'99: Computer Supported Collaborative Learning*, Standford, USA. Also appeared as a preliminary version at AIED99 Workshop on Ontologies for Intelligent Educational Systems, July 1999, Le Mans, France.

Hewitt, J. and Scardamalia, M. (1996). Design principles for the support of distributed processes. In *Symposium on Distributed Cognition: Theoretical and Practical Contributions, at the Annual Meeting of the American Educational Research Association*, New York.

Honebein, P. C., Duffy, T. M., and Fishman, B. J. (1991). Constructivism and the design of learning enivronments: Context and authentic activities for learning. In *NATO Advanced Workshop on the Design of Constructivist Learning Environments*.

Jarke, M., Gallersdörfer, R., Jeusfeld, M., Staudt, M., and Eherer, S. (1995). Conceptbase - a deductive object base for meta data management. *Journal on Intelligent Information Systems*, 4(2):167 - 192.

Kafai, Y. and Resnick, M., editors (1996). *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World*. Lawrence Erlbaum Associates.

Kay, J. and Kummerfeld, B. (1997). User Models for Customized Hypertext. In Nicholas, C. and Mayfield, J., editors, *Intelligent hypertext: Advanced Techniques for the World Wide Web*, LNCS Vol. 1326. Springer.

Kay, J. and Kummerfeld, R. (1994). An individualised course for the C programming language. In *Proc. of the 2nd International World Wide Web Conference*, Chicago, USA.

Lamon, M., Chan, C., Scardamalia, M., Burtis, P. J., and Brett, C. (1993). Beliefs about learning and constructive processes in reading: Effects of a computer supported intentional learning environment (CSILE). In *Annual Meeting of the Americal Educational Research Association*, Atlanta.

Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. (1990). Telos: A language for representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4).

Nejdl, W. and Wolpers, M. (1999). KBS Hyperbook - a data-driven information system on the web. In *Eighth International World Wide Web Conference*, Toronto, Canada.

Nejdl, W., Wolpers, M., and Capelle, C. (2000). The RDF Schema Specification Revisited. In *Workshop Modellierung 2000*, St. Goar, Germany.

Nissen, H., Jeusfeld, M., Jarke, M., Zemanek, G., and Huber, H. (1996). Requirements analysis from multiple perspectives: Experiences with conceptual modeling technology. *IEEE Software*, 13(2).

Papert, S. (1993). *The Children's Machine - Rethinking School in the Age of the Computer*. Basic Books, New York.

Pearl, J. (1990). Bayesian decision methods. In Shafer, G. and Pearl, J., editors, *Readings in Uncertain Reasoning*. Morgan Kaufmann.

Rosis, F. D., Pizzutilo, S., Russo, A., Berry, D. C., and Molina, F. J. (1992). Modeling the user knowledge by belief networks. *User Modeling and User Adapted Interaction*, 2:367-388.

W3C Working Group (1998). W3C resource description framework. http://www.w3.org/TR/PR-rdf-schema/.

Weber, G. and Möllenberg, A. (1995). ELM programming environment: A tutoring system for lisp beginners. In Wender, K., Schmalhofer, F., and Böcker, H.-D., editors, *Cognition and Computer Programming*. Ablex Publishing Corporation.

Weber, G. and Specht, M. (1997). User modeling and adaptive navigation support in WWW-based tutoring systems. In *Proceedings of the Sixth International Conference on User Modeling, UM97*, Sardinia, Italy.