

Extendible Adaptive Hypermedia Courseware: Integrating Different Courses and Web Material

Nicola Henze and Wolfgang Nejdl

Institut für Technische Informatik
Rechnergestützte Wissensverarbeitung
University of Hannover
Appelstraße 4, D-30167, 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 content - 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 integration of hypermedia courses and web material into existing, adaptive hypermedia courses. We show a possible solution which we have used for an undergraduate course about Java programming. We then discuss this solution as well as advantages and problems, and identify several research issues which have still to be solved for adapting distributed course materials.

1 Introduction

While the delivery of instruction, courses and tutorial help is often managed over the web, existing adaptive hypermedia systems have so far neglected the issue of integrating Web material directly into their courses, though most courses can strongly benefit from integrating additional materials into the curriculum. Additional material can provide alternative views or explanations of a topic, it can relate to background information, it can show further developments, etc.

Within our KBS Hyperbook project, we are working on concepts and techniques for building adaptive hypermedia systems which are *open*, e.g. which are able to integrate distributed information resources. In this paper, we will discuss advantages and problems of creating such an open, adaptive hypermedia system. We discuss our approach for building an open courseware system and show how we used it to implement an undergraduate course about Java programming. We then discuss advantages and problems of our current system and identify several research issues which have still to be solved for adapting such distributed course materials.

2 Modeling a course by using a Hyperbook

In our KBS Hyperbook project [11] we are working on a tool for modeling, organizing and maintaining adaptive, open hypermedia systems on the WWW. The adaptive component of the hyperbook personalizes information according to the user's needs, goals and knowledge [8, 9]. For a comparison of the KBS Hyperbook system in the context of adaptive hypermedia systems, we refer to the more thorough discussions in [1, 7]. In this paper we concentrate on the aspect of extendability of such a system, and how different course material can be included and adapted in one integrated system, an aspect not explicitly present in existing systems.

Explicit modeling of structure and contents is important for building reusable, extendible hypermedia applications. The KBS Hyperbook system structures and displays hypertext materials based on a conceptual model. This conceptual model describes courses, different kinds of materials (such as projects, examples, portfolios, HTML pages) and the integration of other information from the World Wide Web. In this section, we first describe the basic structure of our current conceptual model, consisting of different courses and lectures (section 2.1) and then describe the basic *content map* which we use for indexing course material material (section 2.2).

2.1 Modeling the basic structure

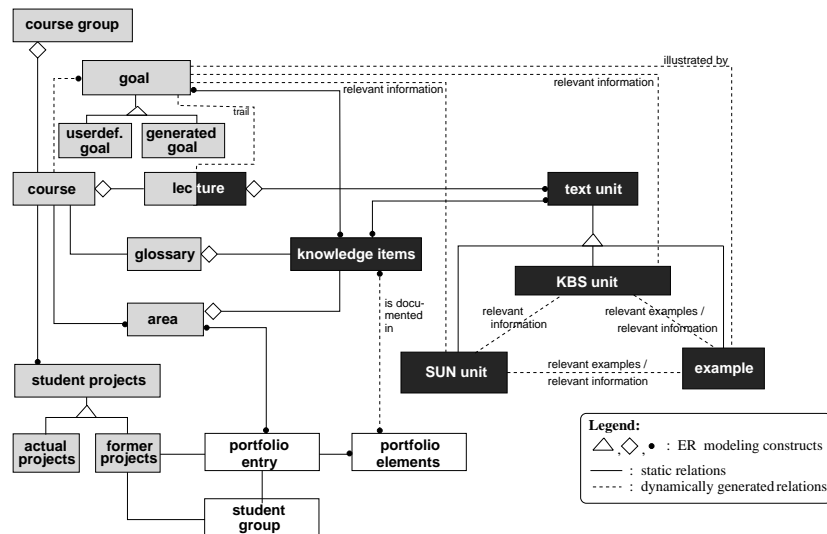


Fig. 1. Conceptual model of the Java hyperbook, course related modeling is highlighted in light grey, modeling of different information resources is highlighted in dark grey

The conceptual model depicted in Figure 1 contains an entity *course* which represents a real course given at some university or at other institutions, and is related to various other entities structuring the course. The concepts belonging to this basic structure are highlighted in figure 1 with light grey.

Each course consists of several *lectures*, which themselves consist of some sequence of *text units*. Several courses can belong to one *course group* (figure 1), this course group integrates different courses on the same topic. Currently, our hyperbook system contains one course group, a CS1 course (Computer Science 1, an introductory course to Java programming). In the last semester (1999 / 2000), we held this course in Hannover for undergraduate students of electrical and computer engineering. A second course, using much of the same material, was given at the University of Bozen, Italy, with our participation. Both CS1 courses are modeled as courses, and belong to the course group “CS1”.

Each course has its *glossary* and a number of *areas* which structure the application domain. Embedding projects and project portfolios in our course materials is an important part of our teaching concept [10]. To model this integration, each course is related to *projects* (figure 1). These projects can be the *actual projects* of the current course, or they can be *former projects*, which give examples of projects performed by students of previous courses and contain the corresponding project portfolios (see section 3.2). 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*). For the selection and the support of goals in KBS Hyperbooks, see [9, 7].

Figure 2 gives an example of the CS1 course given in winter semester 1999 / 2000. The relations mentioned above from a course to other concepts are displayed as links in the left frame. The picture shows specific lectures of this course, current student projects, the different areas of this course, examples of former projects, reference to the next reasonable learning goal, and the reference to the lecture group.

2.2 Enabling Adaptation: Indexing Material

To relate the different types of text units, to support student’s goals, or to provide guidance (i.e. to enable the different adaptation features of the KBS Hyperbook system), we index the course materials which is contained in or should be integrated into the hyperbook. In our conceptual model, we can see the index concepts (called *knowledge items*), and their relations to the glossary, areas, portfolios, goals, and text units (see the relations between the concept *knowledge item* and other concepts in figure 1). Each of these concepts is indexed with a set of such knowledge items. The origin of an information resource is not relevant for indexing, only the content defines the index. We assume that any information resource which should be displayed in our hyperbook is contained on some HTML page. We can therefore define a *content map* which represents the index for an information resource:

Definition 1 (Content Map). *Let $\mathcal{S} \neq \emptyset$ be the set of all knowledge items, and let \mathcal{H} be a set of HTML pages. Then*

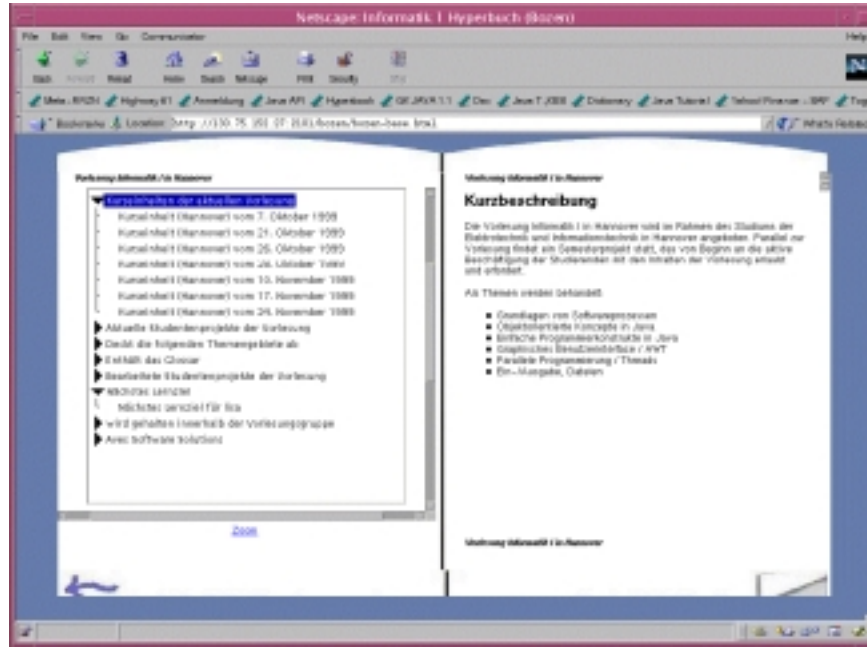


Fig. 2. Example for a course modeled in the KBS Hyperbook system

$$I : \mathcal{H} \rightarrow \mathcal{P}(\mathcal{S}) \setminus \{\emptyset\} \quad (1)$$

is the content map, which gives for each information resource in \mathcal{H} the index of this HTML page, e.g. the set of knowledge items describing its content.

Currently, our content map is built manually by the author of that information resource. We use knowledge items for indexing each kind of information material, which, as we will see in the next section, can originate from our local filesystem or can be located anywhere in the WWW. This kind of indexing is slightly different to other indexing concepts in student and user modeling. In many other adaptive hypermedia systems, dependencies like prerequisites or outcomes are directly modeled within the information resources themselves (see for example [2, 12, 3, 4]). In our system we separate knowledge and knowledge dependencies from the actual information, as we model learning dependencies solely on the set of knowledge items of a hyperbook. The connection between the student modeling component and the hyperbook system is provided by the content map (see definition 1), which maps each information resource to a set of knowledge items. This separation allows easier integration of additional resources on the same topic, then the integrated approach in other systems.

3 Integrating Additional Information Resources

Calling the KBS Hyperbook system an open hypermedia system means, that is able to integrate different sources of materials. On the one hand, it is easily able to refer to and integrate single information resources located anywhere in the web. This will be discussed in section 3.1, where we also show, how we integrated different material in our hyperbook for the Hannover and Bozen courses. In section 3.2 we show how we enlarge the conceptual model to integrate the results of student work, especially the results of the projects they have worked on while learning with the hyperbook.

If the material to be integrated is structured already, integration is still possible but with more difficulties as we discuss in the last part of this paper.

3.1 Integrating *text units*

Each lecture consists of a sequence of *text units* which are used by the teacher during the lecture (see dark grey-colored concepts in figure 1). A text unit can be a *KBS Unit*, which is an information page on our local filesystem. It can be an *example*, which illustrates the use of some concept. As the KBS Hyperbook system allows to integrate arbitrary WWW pages, text units can also be information pages located elsewhere, in our case in the online SUN Tutorial, the pages of which are modelled by the concept *Sun Unit*. If we want to integrate additional information resources, we model them as additional subclasses of *text unit* in our conceptual model.

The paper discusses our current CS1 hyperbook system, which integrates pages from the Sun Java Tutorial [5] into the hyperbook. The Sun Java Tutorial is freely available on the internet and thus very suited for being integrated into the learning material of the Java hyperbook.

For each type of text units, links to related information are computed by the hyperbook system. For example, from a KBS Unit, links to relevant examples are computed, as well as links to relevant Sun Units, which give alternative descriptions of these concepts. The computation of relevant information is done by selecting those information pages whose index (by means of the content map) is not disjunct to the index of the current page. In figure 3, we see on the right hand side the (local KBS) 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 KBS Unit "Methoden" is contained in a specific lecture, we also display the link to this lecture.

In figure 4, we see the integration of a Sun Unit into the hyperbook. The page itself is displayed in the same manner as the local pages. We stream such pages without any modifications into the hyperbook. Thus, links contained on the 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. For this Sun Unit, a link to an example as well as to a KBS Unit is computed.

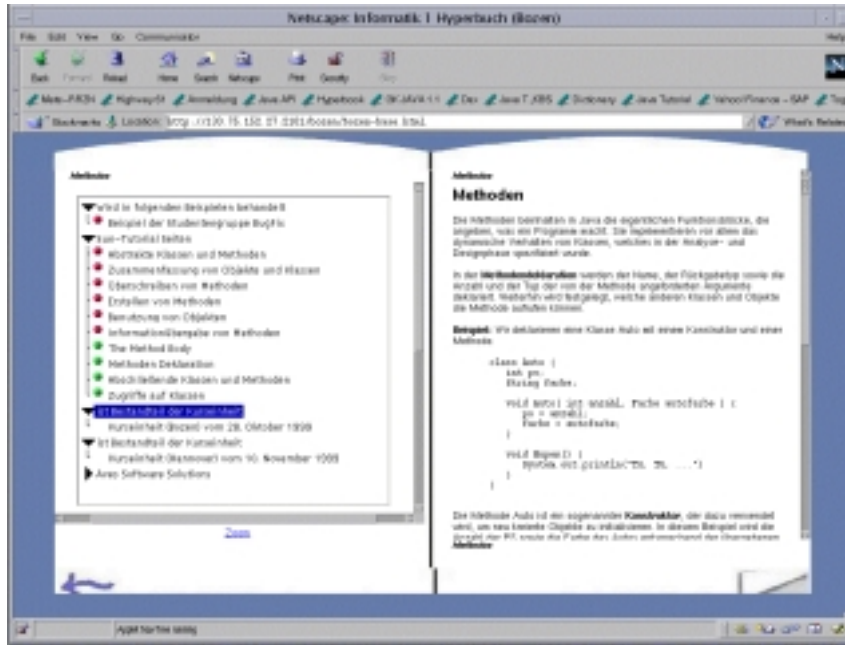


Fig. 3. KBS Unit “Methoden” with links to examples, Sun Units and to the two lectures where it occurs.

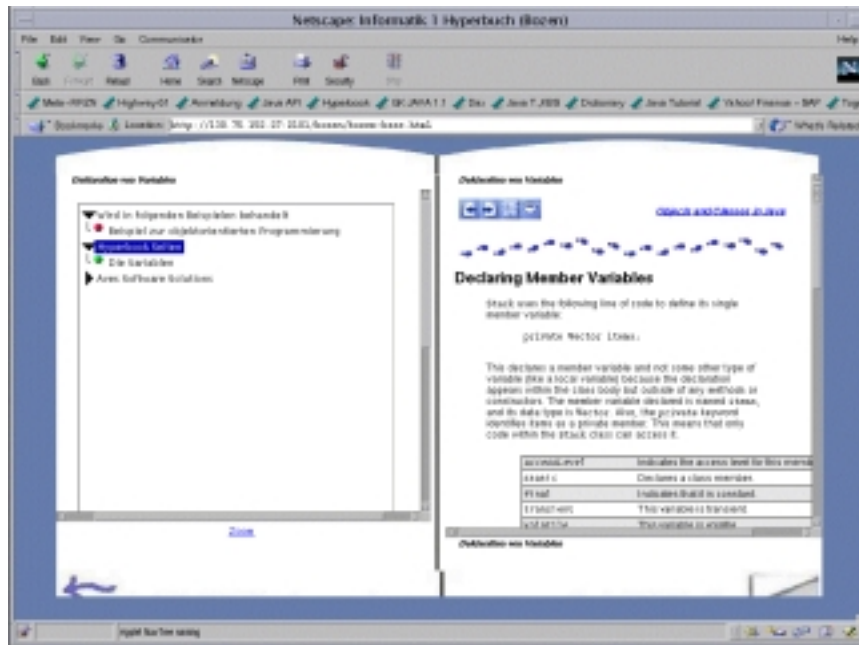


Fig. 4. Example of the integration of Sun Units in the KBS Hyperbook system

All information resources are equal in the sense that they only need to be indexed to be integrated in a particular hyperbook, as described in the previous section 2.2. All these additional information resources are fully integrated and can be fully adapted (just as local pages) to the student's needs: We can propose programming examples described on additional WWW pages, compute reading sequences which contain both local and Web material, calculate the educational state of WWW pages according to the student's actual knowledge state, etc. In the current paper, we will not further describe the functionality of the adaptation component and refer to [8, 7, 11].

This makes clear, that the use of a separate knowledge model makes the hyperbook system robust against changes and extendible. 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 it would be necessary if knowledge, and thus reading or learning dependencies, would have been coded in the material itself.

3.2 Integrating Student Projects: *portfolios*

In order to support project-based learning as described for example in [10], our conceptual model contains the concept *portfolio* (see figure 1). As discussed in [6], 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.

Thus, a portfolio can be modeled 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).

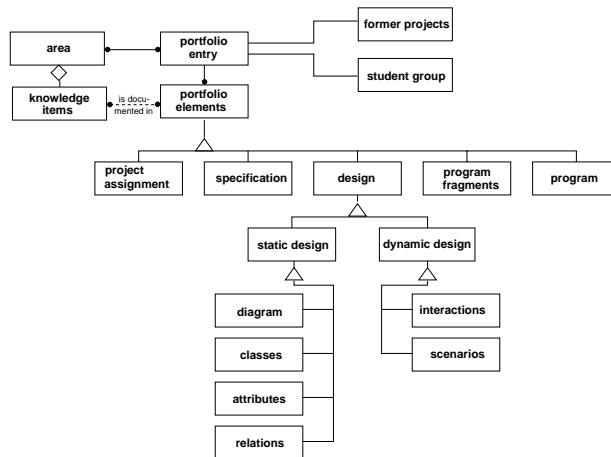


Fig. 5. Schematic view of the portfolio concepts

The portfolio elements represent the different 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. 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	user interface
*specification	event model
static design	*event source
*classes	*event listener
*attributes	adapter
*relations	*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).

4 Research Issues for Distributed Adaptive Systems

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 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 sections.

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 require 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. In subsequent work, we will probably switch to using several knowledge models in parallel, to allow the integration of course materials which use different assumptions about learning dependencies.

Generating Reading Sequences Curriculum sequencing is an important adaptational functionality in hypermedia systems. In case of our *open* hypermedia system, we have sometimes 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 “standalone 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 pre-selection. 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 this 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 standalone 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)¹. 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 really access to the full content

¹ Personal Discussion with Peter Brusilovsky

of the Web (at least to those parts which are relevant and also are annotated with appropriate metadata information).

References

1. P. Brusilovsky. Adaptive and intelligent technologies for web-based education. *KI-Themenheft*, 4, 1999.
2. P. Brusilovsky, E. Schwarz, and G. Weber. ELM-ART: An intelligent tutoring system on world wide web. In C. Frasson, G. Gauthier, and A. Lesgold, editors, *Intelligent Tutoring Systems (Lecture Notes in Computer Science, Vol. 1086)*, pages 261–269, Berlin, 1996. Springer.
3. P. Brusilovsky, E. Schwarz, and G. Weber. A tool for developing adaptive electronic textbooks on WWW. In *Proceedings of WebNet'96 - World Conference of the Web Society*, Boston, MA, USA, June 1996.
4. L. Calvi and P. de Bra. Improving the usability of hypertext courseware through adaptive linking. In *The Eighth ACM International Hypertext Conference*, Southampton, UK, April 1997.
5. M. Campione and K. Wallrath. *The Java Tutorial*. Addison Wesley, 2nd edition, 1999. <http://www.javasoft.com/docs/books/tutorial/index.html>.
6. R. A. Duschl and D. H. Gitomer. Epistemological perspectives on conceptual change: Implications for educational practice. *Journal of Research in Science Teaching*, 26(9):839–858, 1991.
7. N. Henze. Adaptive hyperbooks: Adaptation for project-based learning resources, 2000. Submitted as PhD Thesis to University of Hannover.
8. N. Henze and W. Nejdl. Adaptivity in the KBS hyperbook system. In *2nd Workshop on Adaptive Systems and User Modeling on the WWW*, Toronto, Canada, May 1999.
9. N. Henze and W. Nejdl. Bayesian modeling for adaptive hypermedia systems. In *ABIS 99, 7. GI-Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, Magdeburg, Sept. 1999.
10. N. Henze, W. Nejdl, and M. Wolpers. Modeling constructivist teaching functionality and structure in the KBS hyperbook system. In *CSCL'99: Computer Supported Collaborative Learning*, Stanford, USA, Dec. 1999. Also appeared as a preliminary version at AIED99 Workshop on Ontologies for Intelligent Educational Systems, July 1999, Le Mans, France.
11. W. Nejdl, M. Wolpers, and C. Capelle. The rdf schema specification revisited. In *Workshop Modellierung 2000*, Apr. 2000. To appear.
12. G. Weber and M. Specht. User modeling and adaptive navigation support in WWW-based tutoring systems. In *Proceedings of the Sixth International Conference on User Modeling, UM97*, Sardinia, Italy, 1997.