Adaptive Hyperbooks: Adaptation for Project-Based Learning Resources

Vom Fachbereich Mathematik und Informatik der Universität Hannover

zur Erlangung des Grades

Doktor der Naturwissenschaften

Dr. rer. nat.

genehmigte Dissertation

von

Dipl.-Math. Nicola Henze

geboren am 25. November 1968 in Hannover

2000

Referent: Prof. Dr. W. Nejdl

Korreferent: Prof. Dr. U. Lipeck

Tag der Promotion: 28. April 2000

Datum der Veröffentlichung: Mai 2000

Contents

Summary

Recently, hypermedia systems have become more and more popular as tools for user driven access to information. Adaptive hypermedia systems bring together ideas from hypermedia systems and intelligent tutoring systems, and enable personalized access to information. The KBS hyperbook system is a framework for modeling, organizing and maintaining such adaptive hypermedia systems.

This thesis describes the concept and realization of the adaptation component of the KBS hyperbook system. The adaptation component guides the student through the information space by showing next reasonable learning steps, by selecting projects, generating and proposing reading sequences, annotating the educational state of information, and by selecting useful information, based on a user's actual goal and knowledge.

Learning with hypermedia systems usually takes place in distance learning scenarios. To enable active learning, the KBS hyperbook system follows a constructivist pedagogic approach, building on project-based learning, group work, and discussion. A special focus of the adaptation component is therefore the support of project-based learning and the integration of projects in the curriculum of the user's work with the hyperbook.

The adaptation component uses an indexing concept for describing the content of the various information resources. This indexing concept is also taken as a base for constructing a model of the application domain. A Bayesian inference mechanism calculates estimations about the user's knowledge on top of this domain model.

With the aid of the described adaptation component, intelligent user support can be enabled in hypermedia systems. Furthermore, the system can serve as a development and experimentation framework for further research in the KBS hyperbook system, and in the area of educational and adaptive hypermedia systems.

Keywords: adaptive hypermedia systems, user modeling, constructivism

Zusammenfassung

Dem Einsatz von Hypermedia-Systemen als Werkzeuge zum benutzergesteuerten Informationszugriff kommt in letzter Zeit wachsende Aufmerksamkeit zu. Adaptive Hypermedia-Systeme vereinigen wesentliche Ideen aus den Gebieten der Hypermedia-Systeme und der Intelligenten Tutoriellen Systeme und ermöglichen personalisierten Informationszugriff. Das KBS Hyperbook System ist eine Umgebung zur Modellierung, Erstellung und Verwaltung derartiger Systeme.

Diese Arbeit beschreibt Konzept und Realisierung der Adaptionskomponente des KBS Hyperbook Systems. Die Adapationskomponente führt oder begleitet den Studierenden durch die Informationsvielfalt, indem sie gemäß seinem Kenntnisstand und seiner Ziele vernünftige Lernschritte aufzeigt, geeignete Übungsprojekte ermittelt, eine günstige Lesereihenfolge vorschlägt und auf relevante Informationen verweist.

Zur Unterstützung aktiven Lernens folgt das KBS Hyperbook System einem konstruktivistischen Ansatz, der auf projektbasiertem Lernen, Gruppenarbeit und Diskussion aufbaut. Die Adaptionskomponente unterstützt dabei vor allem projektbasiertes Lernen und die Integration von Projekten in den Lernweg des Studierenden.

Zur Beschreibung der Inhalte verschiedenartiger Informationsquellen verwendet die Adaptionskomponente ein Indizierungskonzept, das auch die Grundlage für die Konstruktion eines Modells der Anwendungsdomäne ist. Innerhalb dieser Domäne schätzt ein Bayesscher Schlußmechanismus das Wissen des Studierenden ab.

Es zeigt sich, daß die dargestellte Adaptionskomponente eine intelligente Anwenderunterstützung in Hypermedia-Systemen ermöglicht. Überdies kann das System als Entwicklungs- und Experimentier-Plattform für weitergehende Forschungen auf dem Gebiet der Lernsysteme und der adaptiven Hypermedia-Systeme dienen.

Schlagworte: Adaptive Hypermedia Systeme, Benutzermodellierung, Konstruktivismus

Chapter 1

Introduction

Books, or, in former times, papyrus, leather, or slabs [27], have been the favourite holder of information since the invention of writing. The advantages of collecting information in books are various: information is grouped together, and often there is a red thread from the beginning to the end.

You can learn by reading a book. Or by discussing with a human. The main difference in these scenarios is, that a human partner will take the special learner into account and will adapt the learning speed and the depth of information to the vis–à–vis. A conventional book cannot adapt itself to the particular needs of its readers – written and printed once, it remains static. If we think of books that are personally written for us, we can suppose lots of useful ideas. For example, the books should not be boring by telling us things we already know or we are not interested in. Think of textbooks, which demonstrate difficult topics by giving examples related to our favourite hobbies. Think of textbooks which give a solid foundation of some topic and refer always – even after years – to the actual research in this area. Imagine a textbook which draws attention to itself if it has new and relevant information for you and provides explanations tailored to your actual knowledge state.

Since the emergence of the World Wide Web (WWW) in 1991 [4], 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 increase 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. 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 documents to the learner's goals and knowledge.

Promising approaches in research come from the area of adaptive hypermedia systems [6]. Adaptive hypermedia systems combine hypermedia systems with intelligent tutoring systems to adapt the systems to the particular users.

This work is a step towards intelligent textbooks. We propose concept and realization of an adaptation component for an open, adaptive 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.

The first chapters outline the scope of this work focussing on adaptive hypermedia systems (chapter 2) and their applications for learning (chapter 3). At the end of chapter 3, we propose a list of requirements that should be fulfilled by adaptive hyperbooks used for education. Since we are interested in building open hypermedia systems on the WWW, it follows that the adaptation of hyperbooks must be sufficiently general for integrating and adapting arbitrary material from the internet. The general functionality of the KBS hyperbook system and its modeling approach is proposed in chapter 4.

The following three chapters describe the adaptation component in detail: chapter 5 introduces the main ideas of the adaptation approach. The indexing concept underlying the approach is discussed, its use is demonstrated, and the applicability is shown. The use of the index is twofold. First, it classifies content of information. Second, the index serves as a base for describing the application domain and thus for describing knowledge. For making inferences about the user's knowledge and for updating the hyperbook due to the user's changing knowledge state, we use a Bayesian inference mechanism. For improving performance of Bayesian inference we propose a clustering algorithm tailored to our application domain (chapter 6). The adaptation functionalities enabled by our general approach are described in chapter 7. Conclusion and outlook on further work accomplish the thesis.

Chapter 2

Adaptive Hypermedia Systems

> *"When the user is building a trail, he names it, inserts the name in his code book, and taps it out on his keyboard ... Several years later .. Tapping a few keys projects the head of the trail. A lever runs through it at will, stopping at interesting items, going off on side excursions"*

Vannevar Bush, 1945 [11]

2.1 Hypertext and Hypermedia

Vannevar Bush's visionary ideas of a kind of mechanized private archive and book collection which allows quick access to its content is often seen as the first step towards the idea of hypertext [90]. The ideas of Vannevar Bush describe a personal archive in which associative connections between documents are possible.

In the AUGMENT project 1962, Douglas Engelbart [26] developed tools for enhancing the efficiency of work. The term *hypertext* was invented 1965 by Ted Nelson [87] in context with the XANADU project. Ted Nelson envised a world wide "docuvers" whose users would reuse material, adding links and annotations.

For the scope of this thesis, we will give a very general definition of hypertext [96]. Discussions on the definitions of hypertext can be found for example in [19, 37, 40, 77].

Definition 1 (Hypertext) *A set of nodes of text which are connected by links. Each node contains some amount of information (text) and a number of links to other nodes.*

Definition 2 (Hypermedia) *Extension of hypertext which makes use of multiple forms of media, such as text, video, audio, graphics, etc.*

The terms hypertext and hypermedia are often synonymous [77]. In this thesis, we will prefer the term *hypermedia*.

Tools for designing and accessing hypermedia documents are called *hypermedia systems*.

2.2 Adaptive Hypermedia Systems

An *adaptive hypermedia system* enlarges the functionality of a hypermedia system. The aim of these systems is to *personalize* hypermedia systems to the individual users. Thus, each user has an individual view and individual navigational possibilities for working with the hypermedia system.

An adaptive hypermedia system combines ideas from hypermedia systems and ideas from intelligent tutoring systems. It belongs to the group of user adaptive systems, which are, for example, user adaptive interfaces or user model based interfaces. For a discussion about adaptive hypermedia systems in the context of user adaptive systems we refer to [62].

Adaptive hypermedia systems use a model of the user to collect information about his knowledge, goals, experience, etc., to adapt the content and the navigational structure. Let us have a look at an example. For a user with little knowledge it might be useful to read more introductory information before going into detail. However, the same information would not be so interesting for an expert. Here, the choice of the right information at the right time is the task of the user model. To give another example, a tourist information system should consider the abilities and disabilities of its users. If a handicapped user requires the opening hours of the city hall, the system's return should also contain hints to the next parking possibility for handicapped people or to the next public transport station, information about the entrance, etc. (see for example [72, 29, 88, 103]).

Adaptation of hypermedia systems is also an attempt to overcome the "lost in hyperspace problem" (for a discussion, see for example [77]). The user's goals and knowledge can be used for limiting the number of available links in a hypermedia system.

For a definition of adaptive hypermedia systems, we follow the proposal of P. Brusilovsky [6].

Definition 3 (Adaptive hypermedia system) *"By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user."*

The support of adaptive methods in hypermedia systems is advantageous if there is *one* common system which serves *many* users with *different* goals, knowledge, and experience, *and* if the underlying hyperspace is relatively *large* [6].

Typical applications of adaptive hypermedia systems are *educational hypermedia systems* where the user or student has a certain learning goal (which also might be an overall learning goal). In these systems, the focus is on the knowledge of the students, which might vary enormously. The knowledge state changes during the work with the system. Thus, a correct modeling of changing knowledge, a proper updating, and the ability to make the right conclusions on base of the updated knowledge estimations are the critical part in an educational hypermedia system.

Other applications are *online information systems*, or, more particular, *online help systems*. Online information systems are, for example, electronic encyclopediae,

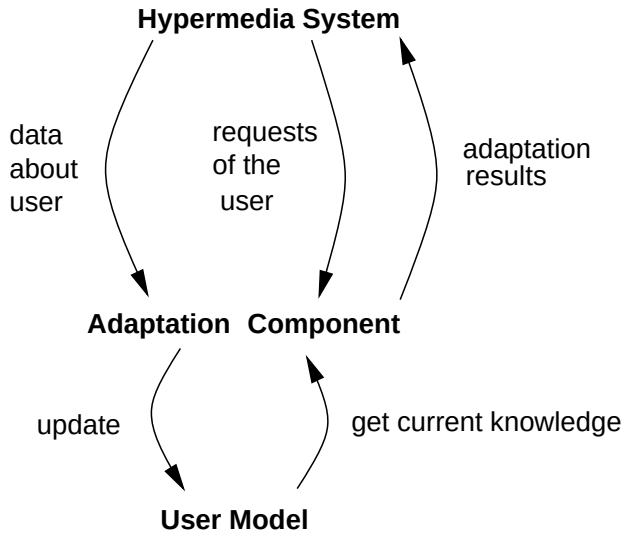

Figure 2.1: Schematic view on adaptive hypermedia systems

document stores, or tourist guides. To select the right information for users with different background or knowledge, these systems also need a model of the user's knowledge. Also the context in which the user requires information from the system is important here: for a quick reference, for elaborating a presentation, or for refreshing knowledge? Online help systems take particular system environments into account, for example the location from which they have been launched (context sensitive help systems).

For limiting navigational possibilities, adaptive hypermedia systems can be combined with information retrieval [64] in *information retrieval hypermedia systems*. Links in such systems are not designed by the author of the system but are based on similarities: a link between two documents is provided if both documents satisfy some similarity condition.

2.3 User Modeling in Adaptive Hypermedia Systems

An adaptive hypermedia system collects information about users. On base of these individual characteristics, it adapts its content and navigational possibilities to the particular user (see figure 2.1).

This section describes general approaches for building user models (section 2.3.1), and several user characteristics (section 2.3.2).

2.3.1 User Models

A user model stores information about the individual user. For a discussion about user modeling, see for example [71, 66]. We can distinguish two main types of user modeling techniques: *overlay modeling* and *stereotype user modeling*.

Overlay Modeling

By overlay modeling [38], the user's state of knowledge is described as a subset of the expert's knowledge of the domain, hence the term "overlay". Student's lack of knowledge is derived by comparing it to the expert's knowledge.

The critical part of overlay modeling is to find the initial knowledge estimation. The number of observations for estimating the knowledge sufficiently must be small. In addition, a student's misconceptions of some knowledge concepts can not be modeled.

Stereotype User Modeling

A stereotype user modeling approach [99] classifies users into *stereotypes*. Users belonging to a certain class are assumed to have the same characteristics. Stereotype classification can be done for each adaptation feature.

When using stereotype user modeling, the following problem can occur: the stereotypes might be so specialized that they become obsolete (since they consist of at most one user), or a user cannot be classified at all.

2.3.2 Characteristics of the User

Adaptive hypermedia systems need data for making assumptions about the user. Brusilovsky [6] identified five features which are taken into account by existing hypermedia systems: user's knowledge, goals, preferences, background, and experience. We claim that also the *learning speed* should be taken into account.

User's Knowledge

In currently existing adaptive hypermedia systems, the knowledge of the user is the most important information for adaptation. Especially in educational systems, the changing of the knowledge state of a user is a critical part for adaptation. The system always has to update its estimation about the user's knowledge, and the adaptation component has to use the actual knowledge state for making its adaptation steps.

User's Goals

Goals of a user depend on his current work with the hypermedia system. Using the system as a reference book requires different adaptation than a more overall learning goal. In educational hypermedia systems, we can distinguish two different types of goals. On the one hand, we have the *overall learning goal* which stretches over the sessions of the user with the system, on the other hand, we have the *problem solving task* which might vary during a session.

User's Preferences

Users of adaptive hypermedia systems have different preferences, for example for font types, pictures, or examples. These are characteristics which could not be estimated by the system without any input from the users. Systems which reflect

the different kinds of preferences let their users tune these features. One can assume that preferences are not subject of rapid change.

User's Background

By a user's background we mean all experiences and knowledge of a user which are not topic of the adaptive hypermedia system itself. For example, programming experience in the language Prolog could belong to the background of a user in a hypermedia system about learning Java.

User's Experience

This characteristic is concerned with the hypertext experience of the user. If the user has worked with a hypermedia system before, has he also worked with an adaptive hypermedia system?

User's learning speed

To my opinion this is also an important user characteristic for educational hypermedia systems and should be reflected. Users with different learning preferences and different learning speed should be supported sufficiently.

2.3.3 Methods and Techniques for Adaptation

As we have seen in section 2.1, hypermedia consists of documents which are connected by links. Thus, there are generally two aspects which can be adapted to the users:

- the content (*content level adaptation*), and
- the links (*link level adaptation*).

Content Level Adaptation

By adapting the content for a user, the document's content is tailored to the needs of the user, for example by hiding too specialized information or by inserting some additional explanations.

According to [6], we can identify the following methods for content level adaptation:

additional explanations Only those parts of a document are displayed to a user which fit to his knowledge or goal.

prerequisite explanations Here the user model checks the prerequisites necessary to understand the content of the page. If the user lacks to know some prerequisites, the corresponding information is integrated in the page.

comparative explanations The idea of comparative explanations is to explain new topics by stressing their relations to known topics.

explanation variants By providing different explanations for some parts of a document, those explanations can be selected which are most suited for the user. This extends the method of prerequisite explanations.

sorting The different parts of a document are sorted according to their relevance for the user.

The following techniques are used for implementing the above stated adaptation methods [6].

conditional text Every kind of information about a knowledge concept is broken into text parts. For each of these text parts, the required knowledge for displaying it to the user is defined.

stretch text Some keywords of a document can be replaced by longer descriptions if the user's actual knowledge requires that.

page or page fragment variants Here, different variants of whole pages or parts of them are stored.

frame based technique This technique stores page and fragment variants into concept frames. Each frame has some slots which present the page or page fragments in a special order. Certain rules decide which slot is presented to the user.

Content level adaptation requires sophisticated techniques for improved presentation. The current systems using content level adaptation do so by enriching their documents with meta information about prerequisite or required knowledge, outcome, etc. The documents or fragments contained in these systems have to be written more than once in order to obtain the different explanations.

Link Level Adaptation

By using link level adaptation, the user's possibilities to navigate through the hypermedia system are personalized.

The following methods show examples for adaptive navigation support.

direct guidance Guide the user sequentially through the hypermedia system. The following two methods can be distinguished:

- **next best** Provide a next-button to navigate through the hypertext.
- **page sequencing or trails** Generate a reading sequence through the entire hypermedia system or through some part of it.

adaptive sorting Sort the links of a document due to their relevance for the user.

- **similarity sorting, prerequisite knowledge** The "relevance" of a link to the user is based on the systems assumptions about the user. Some systems sort links depending on their similarity to the present page. Or by ordering them according to the required prerequisite knowledge.

adaptive hiding Limit the navigational possibilities by hiding links to irrelevant information. Hiding of links can be realized by making them unavailable or invisible.

link annotation Annotate the links to give the user hints to the content of the pages they point to. The annotation might be text, coloring, an icon, or dimming.

traffic light metaphor The traffic light metaphor is the most popular method for link annotation. Here the educational state of a link is estimated by the system due to the user's actual knowledge state. The link pointing to the page is then annotated by a colored ball. A *red* ball in front of a link indicates that the user lacks some knowledge for understanding the pages; thus the page is not recommended for reading. A *yellow* ball indicates links to pages that are not recommended for reading; this recommendation is less strict than in case of a red ball. A *green* ball is in front of links which lead to recommended pages. *Grey* balls give the hint that the content of the corresponding page is already known to the user. Variants in the coloring exist.

traffic lights and hiding A mix of traffic light metaphor and adaptive hiding is also used in some systems. For an evaluation about adaptive hiding and adaptive navigation we refer to [108].

map annotation Here, graphical overviews or maps are adapted with some of the above mentioned methods.

Techniques for link level adaptation are for example discussed in [6] and depend on the specific system. Here the system's assumptions about the user play an important role to decide what and how to adapt.

Link level adaptation restricts the number of links and thus the number of navigational possibilities. It is useful to prevent the user from "getting lost in hyperspace". As in the case of content level adaptation, a description of the content of the documents is required for implementing the adaptation tasks.

2.4 Examples of Adaptive Hypermedia Systems

2.4.1 ELM-ART

The ELM-ART system [9] and its successors ELM-ART II [118] and INTERBOOK [10] are some of the first adaptive hypermedia systems which were used in the WWW. They are based on an the stand alone system ELM-PE [116], an introductory course to programming in Lisp. The authors use an *episodic learner model* [117] (ELM) for diagnosing complete and incomplete problem solutions. The episodic learner model stores knowledge about the user in terms of a collection of *episodes*. These episodes are comparable to *cases* in case-based learning [104]. To construct the learner model, the programming code produced by a learner is analyzed in terms of the domain knowledge on the one hand and a task description on the other hand. This cognitive diagnosis results in a derivation tree of concepts and rules the learner might have used.

In ELM-ART, concepts 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. Static slots contain prerequisite concepts, related concepts, and outcome. Terminal pages contain test slots. Problem pages have a specific slot for storing a description of the programming problem. 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.

2.4.2 INTERBOOK

In the INTERBOOK project [10], electronic textbooks are created on the base 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. The glossary is a visualized domain network. The structure of the glossary resembles the didactical structure of the domain knowledge. Each glossary item corresponds to one of the domain concepts. In addition, each glossary entry provides links to all book sections which use this concept.

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.

INTERBOOK: Adaptive Interfaces

An approach to implement a user adaptive interface for the INTERBOOK project (section 2.4.2) is described in [8]. 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.

2.4.3 PT

PT (personalized text system) is a textbook for learning the programming language C [68, 67]. 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, thus initializes 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.

2.4.4 PUSH

The PUSH project (plan- and user sensitive help) aims to develop and test intelligent help solutions to information seeking tasks [58]. A user can enter questions or follow-up questions, or navigate through the graphical presentation of the application domain. If a user has posed a question, he can manipulate the system's answer by opening and closing subsections, by manipulating the graphics or by clicking on follow-up questions.

The knowledge of the application domain is modeled in a isA-hierarchy. The idea of the system is to provide questions a user might have while reading a document. These questions are concerned with related concepts and knowledge. As the combinatorial construction of all possible questions based on a concept and its relations in the isA hierarchy is too complex and will probably contain too much information, the set of possible questions is limited by the concepts contained on a page *and* the follow-up questions to this concept. The system's implementation is a rule-based approach for queries. As a user verifies each of his tasks by selecting some of the proposed links, the system obtains almost secure information about the user's knowledge.

2.4.5 AVANTI

The aim of the project AVANTI [29, 72] is to develop and evaluate an information system about a metropolitan area for a variety of users with different needs, for example tourists, citizens, travel agency clerks, or users with disabilities.

Sources of information about the users are an initial interview and certain user actions, for example requests for explanations of a technical term. Stereotypes contain assumptions for classes of users. Assumptions from the stereotypes and information about the individual user are used for drawing inferences about the particular user. The results are stored in refined assumptions about the user and serve for better classifying him into one of the stereotypes.

2.4.6 AHA

A free web-course about *Hypermedia Structures and Systems* [18, 12] 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.

2.4.7 OLAE, POLA

OLAE [79] aims to a differentiated and reliable assessment of a student's knowledge in a subdomain of physics. A Bayesian network [92] is constructed for each problem the user is working on. This Bayesian network reflects, among other things, the likelihood that the user would type in particular equations if he would know the corresponding rules. Thus the systems uses a retrospective diagnosis, called *knowledge tracing*. POLA [15] is designed for performing *model tracing*: It can be invoked repeatedly during the problem solving process. POLA constructs the underlying Bayesian network incrementally by adding nodes each time the student performs an observable action.

2.4.8 Plan Recognition in a Multi User Dungeon

The aim of this system [1] is to determine as early as possible which quest a player is attempting in a multi user dungeon (MUD), and to predict which action a player will perform in the next move. Therefore the authors employ keyhole plan recognition, which makes information available to the plan recognizer from non-interactive and often incomplete observations about a user – "as looking into a room through a keyhole".

The system first must learn which actions and positions or sequences of actions and positions tend to lead to a particular quest. This information is modeled in a dynamic Bayesian network [17]. During the testing phase, the dynamic Bayesian network is used to predict the player's quest, next action, and next location.

2.4.9 POKS

POKS [22] is based on a cognitive theory of knowledge structures. It builds a network of implications on knowledge units from a small sample of user data sets, and it uses this induced network to assess the knowledge state with a limited number of observations or questions. The application area is the adapted construction of user interfaces. An overlay model is present.

2.4.10 EPI-UMOD

EPI-UMOD [100] implements a stereotype user model based on Bayesian networks. It uses a separate Bayesian network for each stereotype in which special conditional dependencies between knowledge items are implemented. Each attribute in the stereotype represents the statement that the user knows a particular concept.

2.4.11 HYDRIVE

In HYDRIVE [82] the functionality of the hydraulics systems of an F15 aircraft is taught to technicians and pilots. It focusses on system understanding and trouble shooting strategies rather than on an optimizing action to take at a given point in a problem. The authors use a hierarchical model of the necessary capabilities of the learners and construct a Bayesian network for the entire application scenario. However, this Bayesian network is updated only partially.

2.4.12 KBS Hyperbook System

The aim of the KBS hyperbook system [51, 54] is to build a framework for designing and maintaining open, adaptive hypermedia systems in the internet. It implements project-based learning [104].

In KBS, concepts are related to each other on the base of a conceptual model of the hyperbook. Observations about users are made when a user has performed some project of the hyperbook's project library.

Each hyperbook unit is indexed with some knowledge concepts. A separate knowledge model is constructed, containing the knowledge concepts of the application domain and their learning dependencies. Thus, the hypertext documents itself do not contain any prerequisite or outcome information.

A glossary containing the concepts of the knowledge model is generated. For each glossary item, links to examples, to hyperbook units, and to pages from other electronic books available on the WWW are generated.

KBS uses the traffic light metaphor (see section 2.3.3) for adaptive annotation. A page sequencing algorithm generates reading sequences according to the user's goals and knowledge. For helping the user to find his way through the hyperbook, the system also generates a next learning step for the user by comparing his actual knowledge state with knowledge he should have after finishing the book.

KBS supports explicitly goal-based learning. Users can define their own learning goals or can request the next learning goal from the system. For each of these goals, a reading sequence containing necessary knowledge (prerequisite and actual necessary knowledge) for reaching the goal is generated. In addition, suitable projects are

selected and an information index is presented which contains both documents from the hyperbook itself and documents located anywhere in the internet.

The selection of suitable projects is based on algorithms which reflect both, the prerequisite knowledge necessary to perform the project, and how well the project matches to the user's actual learning goal.

KBS also adapts to the different learning speeds of the users by supporting this kind of goal oriented learning. Users can define how much and what to learn next. If the system observes a user mastering an advanced project sufficiently well then it updates its estimation about this user in relation to the mastered topics as well as to prerequisite knowledge concepts. Thus the user can go on with further, advanced topics. If the system observes that a user is not quite familiar with some topic, it proposes similar examples or projects, which contain only a few amount of new information.

The implementation technique used in KBS is a Bayesian network which operates on the complete model of the application domain. Whenever observations about a particular user are made, they are propagated through the entire network.

A focus of KBS is the extendibility of the system in respect to the World Wide Web. To create *open* adaptive hypermedia systems, the indexing approach chosen in KBS allows to treat each information unit equally independent of its origin. Thus, HTML pages from the World Wide Web can be integrated in the same way as documents stored in the hyperbook's library.

2.5 Discussion

In the following we will compare the KBS hyperbook system to other hyperbook-like approaches.

The approach chosen in PUSH is designed for a different scenario compared to KBS. In addition, PUSH binds adaptation information on the documents themselves, thus its authors use very different implementation strategies.

The application in the AVANTI project is also different compared to hyperbook-like approaches. Since the application domain (information systems) is very different from educational hypermedia systems, the techniques used here are almost incomparable.

The approach chosen in OLAE / POLA focusses on problem solving support. The use of the Bayesian networks in POLA is different to their use in the KBS hyperbook system, since POLA uses smaller networks and employs them for recognizing problem solving strategies rather than for knowledge estimation. The same holds for EPI-UMOD and HYDRIVE. EPI-UMOD uses one Bayesian network for a class of users, while KBS uses one for each user. As HYDRIVE only updates parts of the Bayesian network, different techniques are used there, too.

The application of Bayesian networks for plan recognition is also a very interesting approach. But here, the application – learning the system – is also different. Dynamic Bayesian networks are also used in [103]. As we implement goal-driven learning, we have no time critical tasks (time-critical in the sense of dynamic Bayesian networks). A comprehensive review of current work in using uncertainty management techniques in user modeling is given in [61]. The size and purpose of the Bayesian network used in KBS differs from other approaches using Bayesian net-

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

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

works. KBS is the only system which uses a Bayesian network for modeling and propagating the whole knowledge domain (see section 6).

The most comparable approaches to KBS are AHA, ELM-ART II, INTERBOOK, and PT. Therefore we will compare these systems due to the identified criteria proposed in section 2.3. Figure 2.2 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 or the ANATOM- Tutor [2]. PT assumes, that all users of the system have the same background. Thus it is a general assumption rather than a background characteristic. Content-level adaptation is not done by any of the five hyperbook-like systems. From the above stated examples, only EPI-UMOD uses content level adaptation.

The methods used for navigation support can be seen in figure 2.3.

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 made on base of the knowledge model. The algorithm for page sequencing proposed by INTERBOOK neglects the connections of the pages. Pages are selected if they are recommended to visit and it is not explicitly taken into account if they also fit to the previously read page. The page sequencing algorithm in KBS generates a sequence of pages that reflects a didactical ordering, since it is aware of a wider context (learning goal or information request).

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 be adjusted to the other, existing documents. In addition, KBS is able to select and propose projects and supports the learners work with these projects.

Chapter 3

Teaching

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 only 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 the hyperbooks, 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.

3.1 Specific Situation 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 [23]).

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 [57].

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.

The teacher's role is to coach the student's work and to teach them the initial concepts necessary for their task.

Real-world examples are characterized by the complex context in which they arise: few applications of techniques and concepts occur in the simplified forms used for highly abstracted exercise problems often presented to learners (see also the discussion in [23]). While abstraction is of course necessary and small exercises can be used to discuss specific issues, project-based learning has to be used to rebuild real-world complexity. The global project context determines the learners perspective on a given task, while subtasks in a smaller context provide guidance of the learning process.

The ability to develop multiple and alternative perspectives on a problem is also a central skill for performing tasks in an authentic activity. Collaborative learning promotes the exchange and reflection on different views. As project work is often done in teams, learners can train their capabilities for team-work and collaboration.

3.2 Constructivism as a Theory of Knowledge

Within the last 10 years, constructivism as a philosophical, epistemological and pedagogical approach has found a great deal of attention. While several authors have concentrated on various aspects of this approach, one of the most influential authors is Ernst von Glasersfeld, who discussed *radical constructivism* as a theory of knowledge and cognition (e.g. in [114]) and its applications for teaching (e.g. [113]). In [114], he defined constructivism by the following principles:

- Knowledge is not passively received, neither by sensing nor by communicating, but is actively built up by the cognizing subject.
- The function of cognition is adaptive, and tries to increase fitness or viability. It serves the organization of the experiential world of the subject, not the discovery of ontological reality.

As this characterization is rather oriented towards the knowledge construction of one subject (not explicitly taking into account more social aspects of knowledge construction), various researchers have suggested a more context and socially oriented view of constructivism (see for example the discussion in [14]). A less ambitious definition just acknowledges that learners (including scientists) must construct and reconstruct their own meaning for ideas about how the world works ([39]), concentrating just on the first principle of Glasersfeld definition. Even so, this still leads to a change in the role of the teacher, where (as discussed in [94]) the teacher needs to create situations, where the student can work on useful problems, where the teacher provides counter-examples compelling reflection and reconsideration of solutions, and where the teacher is acting as mentor stimulating initiative and research rather than being a lecturer who transmits ready-made solutions.

In the next chapter we will review a few approaches taken by researchers and educators following a constructivist approach and then proceed to show how conceptual

modeling helps to implement our didactic goals based on such an approach in an introductory computer science course.

3.2.1 Constructivism and Teaching

In [113], Glasersfeld sees constructivist pedagogy as a counterpart to behavioristic pedagogy, and stresses the importance of teaching (which aims at the generation of understanding) versus pure training for performance (often geared at perfectly solving textbook problems). Knowing as an adaptive activity leads to a set of successful or viable concepts, models, and theories relative to a context of goals and purposes. Learning requires self-regulation and the building of conceptual structures through reflection and abstraction, problems are not solved by the retrieval of memorized "right"answers.

In the introduction of [112], Glasersfeld also stresses the need to understand the students' thinking and to encourage them to reflect on their models as a means to improve them (e.g. by verbalizing it). Social interaction is an important stimulus for this reflection as well as for motivating knowledge construction and adaptation.

Constructivist concepts discussed in the papers from [112] include problem-oriented and inquiry-oriented learning and discussion, thought protocols to obtain insight into student's mathematical thinking, the necessity of contradictions for further construction (whose awareness is depending on the previous knowledge), the importance of student models, learning as cognitive restructuring, teaching through problem solving, whole class interactions and small group interactions, curiosity, and reflection.

Papert and his colleagues [91, 65, 98], who use the term *constructionism* to stress learning as a (social) design activity, based on computer science and computer use for learning. Similar to others, they stress that students construct new knowledge with particular effectiveness when they are engaged in personally meaningful products. The goals of the teacher are to engage the learner in active participation, problem solving, interdisciplinary work, reflection, and discussion. They also stress the intrinsic motivation resulting from the learners choosing their own projects, and an open learning community with mentors and students. Though the members of the group focus mainly on the learning of children, the principles of their approach are applicable to student and professional learners as well.

The social and knowledge sharing aspect is stressed in another long running project, the CSILE project (computer supported intentional learning environments [73]) and its successor *Knowledge Forum* (see e.g. [102, 73, 55]), which aims for a networked, collaborative learning environment designed to support a classroom-based knowledge-building community and collaborative knowledge building (modeled after scientific work in a research team). It provides a communal database, which stores notes, annotations and discussion items and links them together in a network of nodes (visualized as a knowledge map). It focuses on *intentional learning*, where learners strive to expand their knowledge collectively.

3.3 Implementing Constructivist Teaching Concepts in a CS1 Course

Based especially on the ideas of learning as design activity (as advocated by Papert and his colleagues [91, 65, 98]) and learning as an intentional activity involving knowledge-building and discussion (as in CSILE [102, 73, 55]), 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 (see section 4.2) 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 then they can ask the hyperbook for the next suitable learning goal.

3.3.1 Project-Based Learning

Constructivist learning models and project-based learning can be supported by different approaches and can be viewed from different perspectives. We will sketch some of these, starting from the concepts discussed in [104] as well as from their learning principles.

Simulation-Based Learning By Doing: Acquisition of knowledge is guided by goals or projects actively pursued by the students. Knowledge and techniques are learned and used to fulfill specific tasks which are needed to reach the project goals. Teachers have to give help when needed. To support learning by doing, simulations of all kinds of tasks can be built.

Incidental Learning: Projects and goals of a course have to be selected by the teacher in such a way that skills and knowledge needed to pursue these goals correspond to the (conventional) course content. Obviously, in a task oriented learning environment not only a single set of techniques leads to project success. Hence, although the base set of knowledge and skills that will be learnt is set, students can individually control their learning, depending on their previous knowledge and their individual preferences.

Learning by Reflection: Students are encouraged to reflect on given problems and on different solutions found by themselves or other student groups. Continuous discussion of teachers and students leads to original solutions and new insights.

Case-Based Teaching: Presentation of knowledge by the teaching staff depends to a certain extent on the progress students make in solving the given problems. Support is oriented mainly around cases with attached related knowledge, facts and problem solving methods. These cases can be continuously added to the knowledge base of the learning environment and represent an increasing knowledge and support base for the students.

Learning by Exploring: Communication between teachers and students is the main part in this approach. Topics of the course are discussed in study groups or with the teacher. Small learning units contain the needed knowledge. Learning units are presented in the knowledge base which is extended by students and teachers. Students are engaged to study and to find out facts, skills, and research results on their own.

3.4 Excursion: The KBS Virtual Classroom Project

Within the *KBS virtual classroom project* [48], we have been working on a virtual learning environment based on internet and the World Wide Web since 1996. One main goal for this project has always been to utilize the full power of these techniques to innovate teaching and learning in our courses, instead of just transplanting ordinary lectures onto the internet.

The KBS virtual classroom project provides the environment for using adaptive hyperbooks for university teaching. It has been evaluated [44, 49, 45]. The results from the evaluations were directly used by every subsequent implementation of both the learning environment and the KBS hyperbook system.

Availability of the Working Environment

In order to make the working environment continuously available, access to all parts of the working environment (including all information and software tools) is enabled via the internet. Access to the internet for students is either from campus computers, via phone lines and terminal servers (administrated by the university computing center and the students themselves), as well as via internet providers.

Most tools are available for all current operating systems, so students can use them locally at every computer they have access to, including of course their home computers. If license restrictions make this impossible (as in the case of a large software engineering tool), at least anytime access over the internet is available (currently with X11 interface). Locally used tools are always internet-based, so access to central servers, repository, and communication facilities is always possible. Most of the online information is also downloadable for offline use.

Project-Based Learning

Courses supported by the KBS virtual classroom environment use projects, tailored specifically to the course contents. As an example, the CS1 course for undergraduates consists of one large programming project which stretches over the whole semester, the software engineering course focussses on one larger project spanning two semesters.

All project results (programs, documentation, project work) are presented on the World Wide Web, where they are available for other groups as well as for students in future semesters. This motivates students to not only solve larger real-world problems, but also elaborate them in a way suitable for external presentation.

Team-Oriented Learning and Mentoring

Project-based learning is done in groups of two to four students. Groups are formed at the beginning of each semester. Students in such a group work on a common project and present their results together. Collaborative learning and working is encouraged. For each group we assign a personal mentor, which can either be a graduate student, a Ph.D. student, or the professor. This mentor is available personally at specific hours during the weeks, or anytime by electronic communication facilities. Group meetings and discussions are possible in personal or electronic form.

Electronic Communication Facilities

Each student group has a *group communication center*, which includes *e-mail lists* to all students within this group (plus another one including the group mentor), a *communication room* on the WWW (read and write access restricted to members of the group), and a *presentation room* on the WWW (which is readable by everyone).

Each course includes three discussion groups (implemented as newsgroups), one for official (*announcements*), one for general discussion and questions as well as course oriented exercises plus student answers (*discussion forum*), and one for free communication not directly related to the course (*cyber cafe*). The announcements and discussion forum are also automatically archived, indexed and made available over the WWW. Synchronous communication at present can take place over a text based chat tool, the KBS *online chat forum*.

All facilities are available on many operating systems, and on computers connected to the internet anyway.

Network Environment

A server located at our institute serves as the central repository for all course-related material like lecture slides, tutorials, web pages, hyperbook, programs, etc. All central repositories (WWW communication and presentation area, etc.) are also stored at the institute's server. A variety of working environments for accessing the data is supported: Classically working on a client at the institute and accessing the data through Ethernet/NFS; accessing data from one's home PC via a modem connection to a dedicated machine in our network; accessing data from the university computer pool or from the (student's) home PC connecting via modem to a dedicated student server, which is configured to support a large number of parallel modem and ISDN connections.

3.5 Discussion: Requirements for enabling project-based learning

It is a central requirement of a constructivist teaching approach to keep the hyperbook as maintainable and extendible as possible. A maintainable structure allows to integrate the students' results seamlessly and to keep the theoretical course materials up to date with minimum effort. The implementation of such a sophisticated conceptual backbone requires rigorous modeling. We propose a *conceptual modeling* approach for hyperbooks (see section 4.2) which explicitly represents all aspects of the hyperbook application domain, and access to information.

For integrating student projects into the hyperbook, we use the idea of *portfolios* as discussed in section 4.2.4. The students demonstrate in the portfolio which concepts they have used in their course project.

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.*

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 lacks 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, selecting algorithms have to be found. They should present the most suitable projects to the user which match to his current learning goal and consider his actual knowledge.

The KBS hyperbook system implements the above stated requirements. In chapter 4 we describe the functionality of the KBS hyperbooks system, its modeling approach focussing on conceptual modeling, integration of projects and portfolios, and the openness of the system for integrating material located anywhere in the WWW.

Chapter 5 and 6 describe the concept and realization of the adaptation component for the KBS hyperbook system in detail. In chapter 7 the adaptation in the KBS hyperbook system is shown.

Chapter 4

The KBS Hyperbook System

The KBS hyperbook system is a tool for modeling, organizing and maintaining adaptive, open hypermedia systems 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.

Adaptive hyperbooks personalize information according to the user's needs and knowledge. Typical applications of these books are educational hypermedia systems, where the system models some course, guides the student through the course and its learning material, and supports the student's access to useful information. The project-based teaching approach we have chosen for creating adaptive hyperbooks for education is described in chapter 3.

This chapter starts with a definition of adaptive hyperbooks and an overview of the modeling approach we apply in the KBS hyperbook system. 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 (Computer Science 1) course, which is an introductory course to programming in Java, given for undergraduate students of electrical engineering and computer science. We show the way we integrate student projects into the hyperbook based on the idea of portfolios.

4.1 Definition of Hyperbooks

Since the emergence of the World Wide Web, the concept of hypertext has become a main representation and presentation format for a variety of applications. Quite prominent among them are hypertext books or *hyperbooks*, which are characterized as "a grouping of electronic texts which can be considered as an entity"[69]. In most cases, these hyperbooks still retain the conventional book structure, and are partitioned into (sub-) documents called chapters, sections, subsections, or appendices [83]. This definition includes *electronic books*, which can be characterized as "existing books meant to be read on a computer screen" [59].

Starting from this simple definition, transfer of printed books into electronic form has emerged as a wide research area [97]. Some of these projects deal with a broad variety of printed books which are translated to data formats such as plain text, ASCII derivates, or the acrobat reader format; the table of contents being trans-

lated to a hypertext interface to the book [5]. More sophisticated approaches re-edit existing printed books by adding pictures, remarks, and annotations as hypertext links [111]. Or they start from a specialized hypertext system and build up networks of documents in a way not possible in printed versions, such as Dickens Web [75] which is based upon the INTERMEDIA system [74] and the applications realized with the ATHENA system [56]. Research has also been done on the design of specialized hardware for these electronic books such as organizers, portable PCs, and electronic book readers [105].

We focus on a definition of hypertext books with particular emphasis on structure, semantic contents, and corresponding functionality of such a book, and use the term *adaptive hyperbook* to distinguish them from other variants of hypertext books.

Definition 4 (Adaptive Hyperbook) *An adaptive hyperbook is an information repository which integrates and personalizes a set of distributed information using explicit conceptual models.*

Research on the development of hyperbooks has focused on the educational sector, where hypertext technologies are used to implement learning environments [81, 68, 10, 19, 25, 110, 51, 54] and intelligent tutors [7, 9]. These systems provide the contents which have been covered so far in normal text books, and integrate them into a hypertext system which guides the users during their learning processes.

4.2 Conceptual Modeling for Adaptive Hyperbooks

Structuring concepts in domain ontologies has been an important activity for example in natural language processing ([28, 78, 70]) and information retrieval ([42, 80, 3]). It is related to work in terminological research (for a comparison see e.g. [36]).

The KBS hyperbook system structures and displays hypertext materials based on conceptual models. We are using ontologies in the wider sense discussed in [41], including conceptual models, representation ontologies, task ontologies, etc., and not just taxonomic domain ontologies, though these domain ontologies also play a role in our hyperbook system, when we want to structure our hyperbook based on its contents. To avoid misunderstandings, we use the term *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, see figure 4.1. As an example we show the integration of the Sun Java tutorial [13] 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 a dialect of the object oriented conceptual modeling language Telos [84], which is implemented in the ConceptBase system [63]. This language combines object oriented concepts

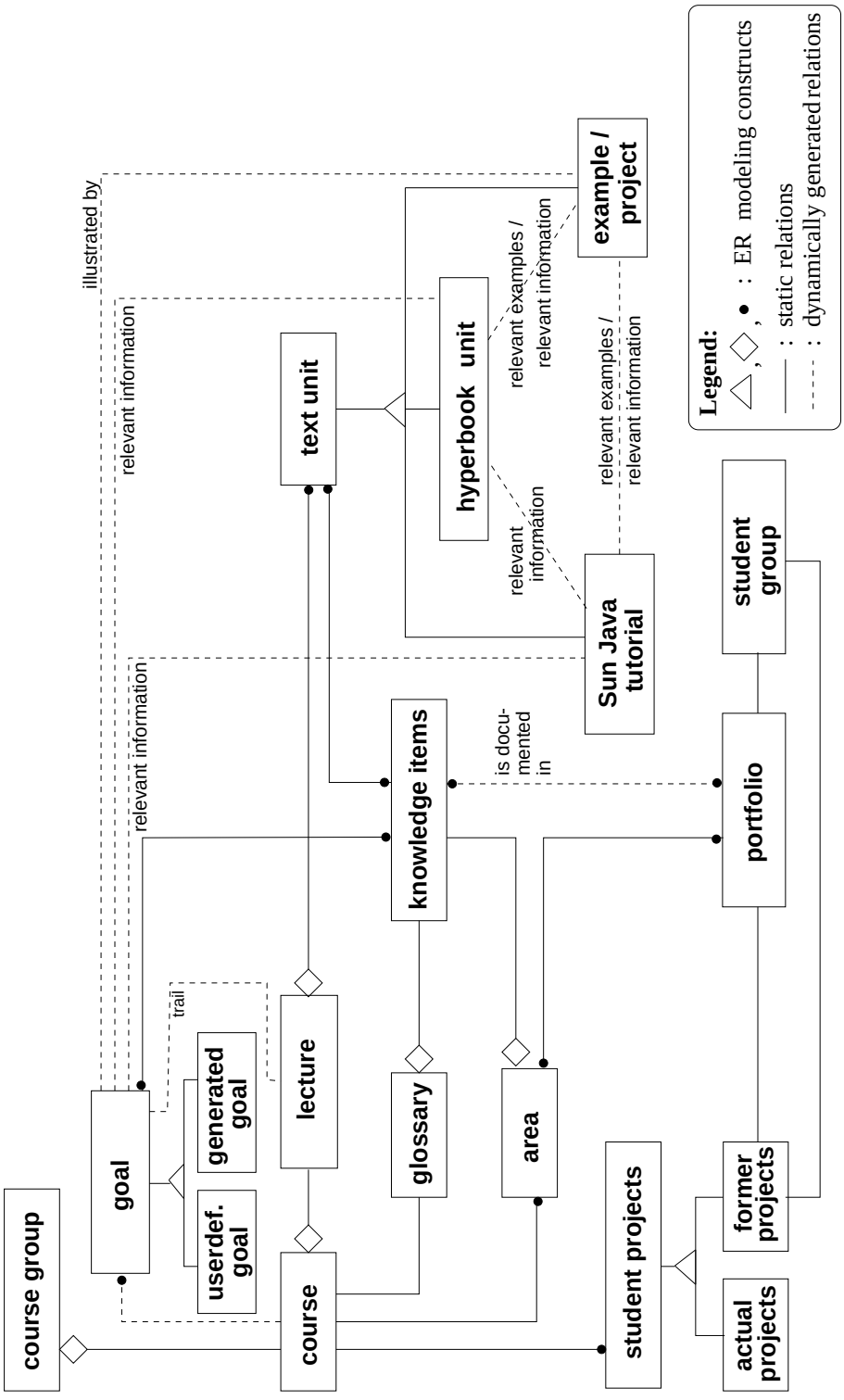

Figure 4.1: Conceptual model of the CS1 hyperbook

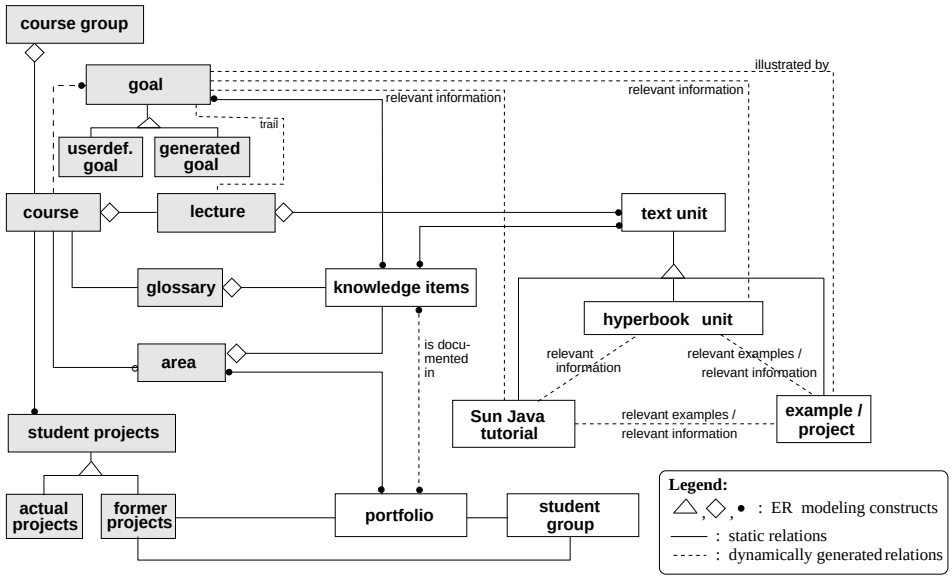

Figure 4.2: Conceptual model of the CS1 hyperbook, course related modeling is highlighted

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. [89]). This allows us to research on meta modeling for the KBS hyperbook system [85, 86] using e.g. ideas and concepts of the Resource Description Framework [115] (RDF).

4.2.1 Modeling courses and lectures

Central for this part of our conceptual model (see the highlighted concepts in figure 4.2) 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 4.2). 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 Europäische Akademie Bozen. 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, see section 7.2.2) and a number of *areas* which structure the application domain (see section 5.3.2).

The embedding of project and portfolios in the learning material is an important part of our teaching concept (see section 3.5). To model the integration, each course is related to *projects* (figure 4.2). 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 (see section 4.2.4) of them.

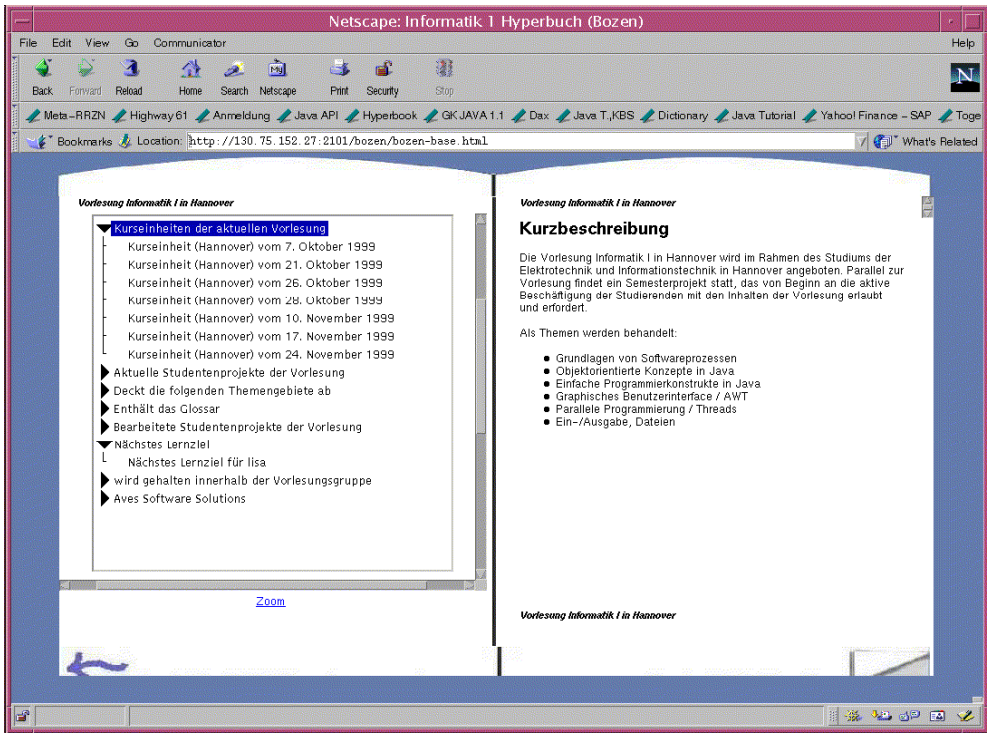

Figure 4.3: Example for a course modeled in the KBS hyperbook system

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*). For the selection and the support of goals, see sections 7.3 and 7.4.

Figure 4.3 gives an example of the CS1 course given in winter semester 1999 / 2000. Each above mentioned relation from a course to other concepts is displayed as 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.

4.2.2 Modeling Different Information Resources

Each lecture consists of a sequence of *text units* which are used by the teacher during the lecture (see highlighted concept in figure 4.4). A text unit can be a *hyperbook unit*, thus an information page belonging to the hyperbook's library. Or it can be an *example* showing the use of some concept. As the KBS hyperbook system allows to integrate 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*.

From each types of these text units, links to related information are generated (see section 7.2.3). For example, from a hyperbook unit, links to relevant examples are generated, and links to relevant Sun tutorial pages, which give alternative descriptions of these concepts. In figure 4.5, 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

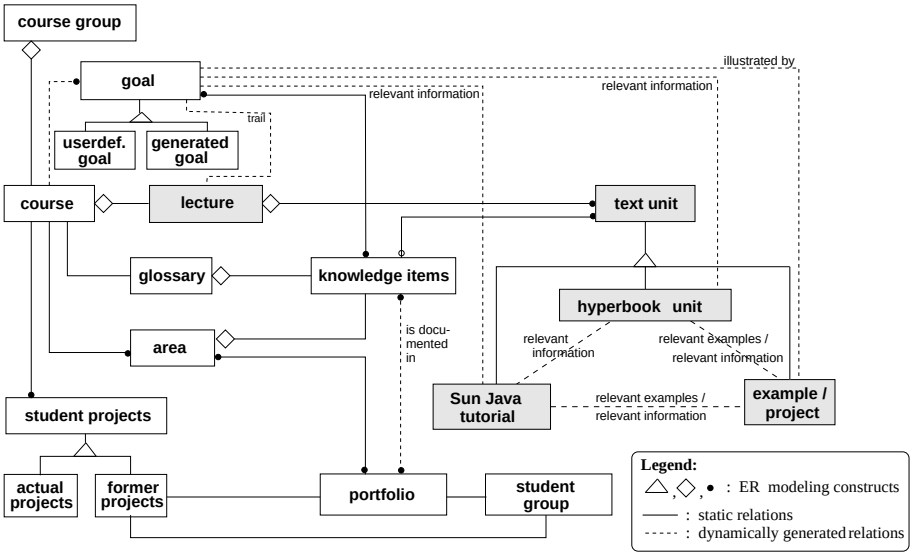

Figure 4.4: Conceptual model of the CS1 hyperbook, modeling of different information resources is highlighted

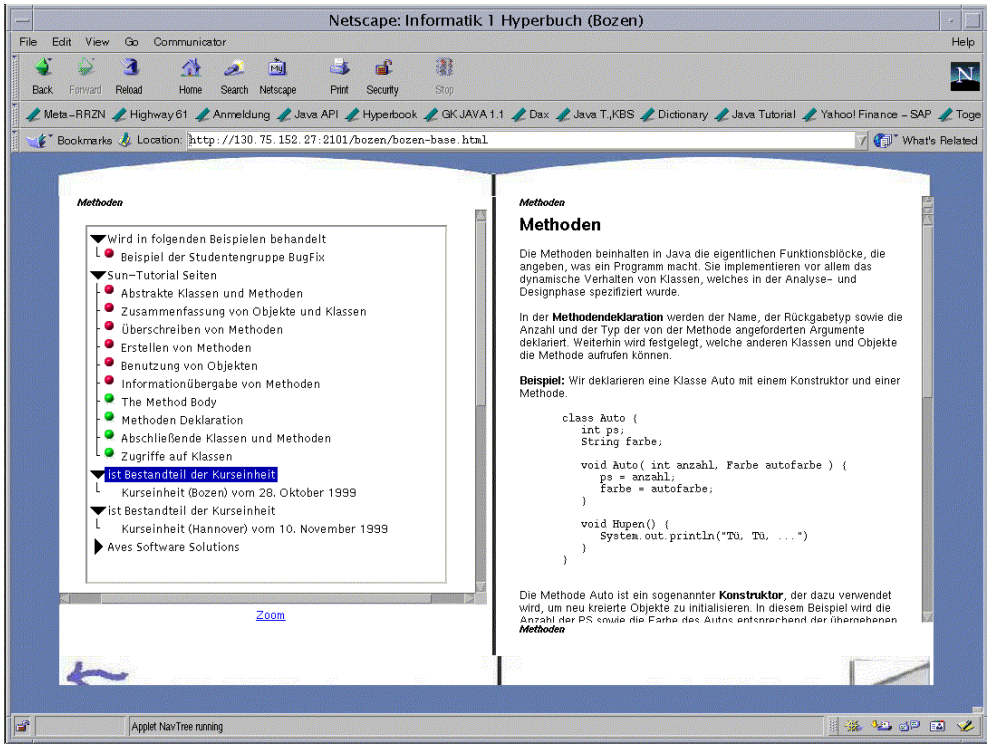

Figure 4.5: Hyperbook unit “Methoden” with links to examples, Sun Java tutorial pages and to the two lectures where it occurs.

the hyperbook unit ”Methoden” is contained in a lecture, we also find a link to this lecture.

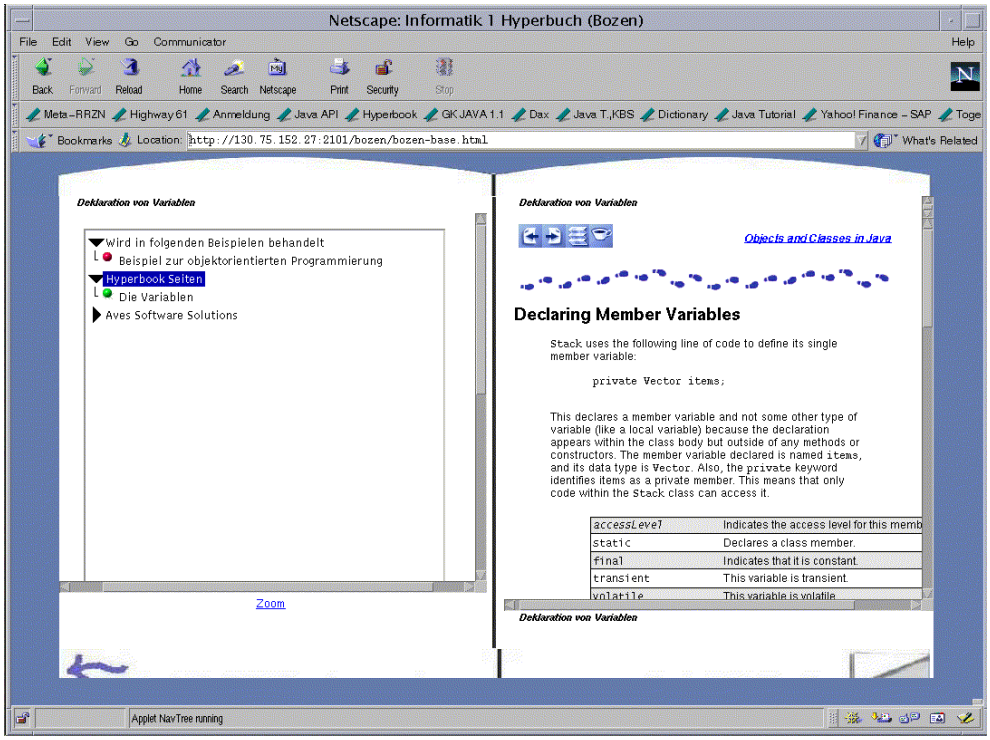

Figure 4.6: Example of the integration of Sun Java tutorial pages in the KBS hyperbook system

In figure 4.6, we see the integration of a Sun Java tutorial page into the hyperbook. The page itself is displayed in the same manner as it would origin from the hyperbook library. 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 tutorial page, a link to an example as well as to a hyperbook unit is generated.

4.2.3 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 chapter 5. In the conceptual model, we see the index concepts, which are called *knowledge items*, and their relations to the glossary, areas, portfolios, goals, and text units (see highlighted concepts in figure 4.7). Each of these concepts is indexed with a set of such knowledge items.

4.2.4 Modeling Portfolios

In order to support project-based learning as described in section 3.5, our conceptual model contains the concept *portfolio* (see figure 4.8). As discussed in [24], assessment

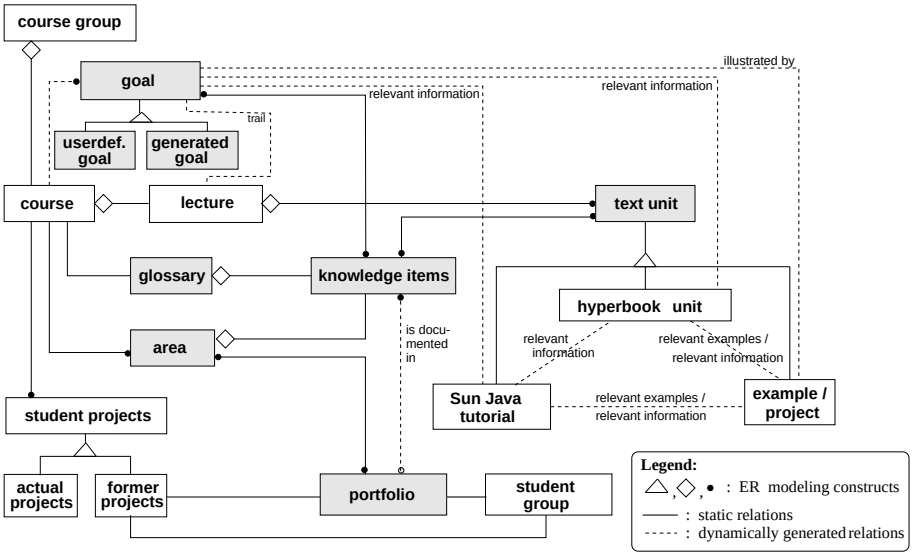

Figure 4.7: Conceptual Model of the CS1 hyperbook with emphasized index strategy

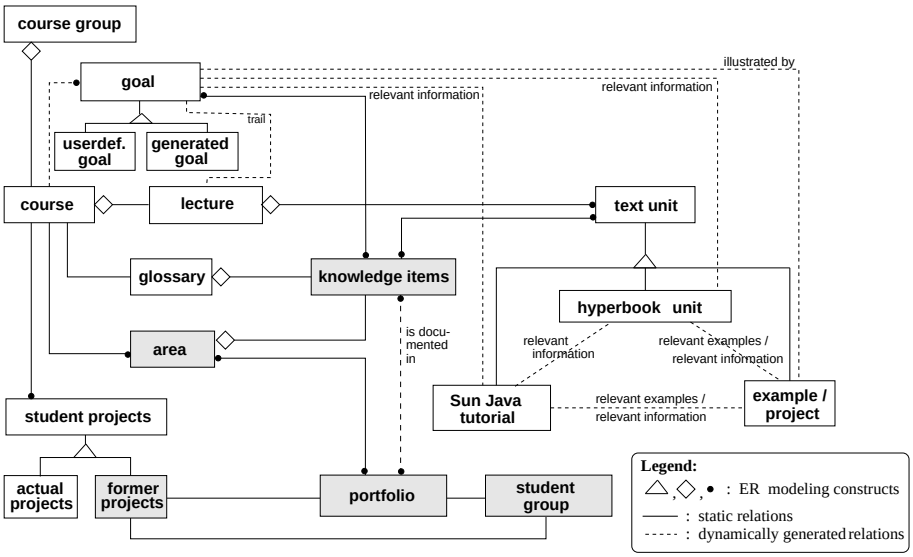

Figure 4.8: Conceptual Model of the CS1 hyperbook, modeling of portfolios is emphasized

based on portfolios realizes the the idea that project results can be used to represent and to assess which concepts a student has successfully applied or learned.

We model a portfolio as a part-whole hierarchy representing the different parts of each student project. In figure 4.9 we see this hierarchy, which enlarges the conceptual model (section 4.2). This hierarchy mirrors the simplified software modeling process we use in our CS1 course. Important parts are the specification written by the students, an object oriented design proposal consisting of several subdocuments, documenting the implementation, and the program code itself. The program code is

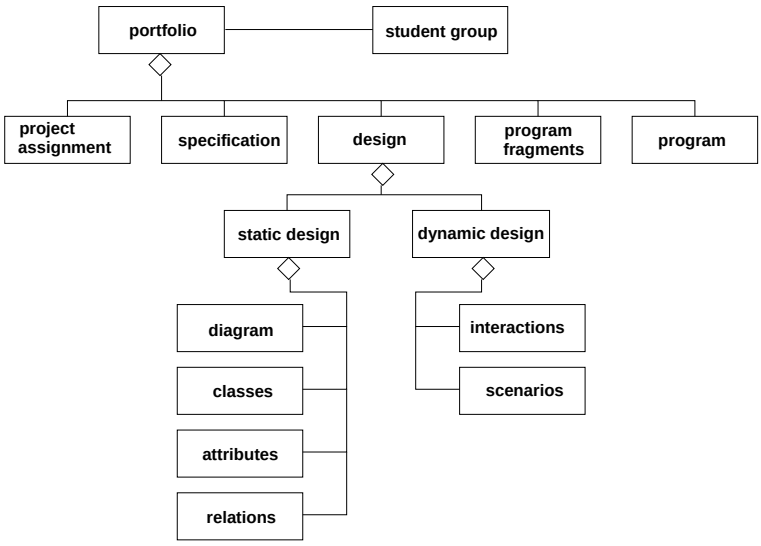

Figure 4.9: Schematic view of the portfolio part-whole hierarchy

broken down into different program fragments, each showing the use of some specific concept of the application domain.

To help the students defining their own portfolios, we propose a list of concepts, containing topics of the Java domain. There are some concepts which are mandatory for their portfolios (for example the scenarios and interactions from the dynamic design of their project), and a lot of other optional concepts. Thus, the students can choose some of these optional concepts for representing their work in the portfolio individually.

A part of portfolio concepts can be seen in the following list. Concepts marked with an asterisk must be contained in a student's portfolio, other concepts are optional. Each concept in the list is indexed by exactly one knowledge item so that we can easily integrate the portfolio concepts of the student group in the hyperbook.

```
object oriented design
 *specification
  static design
    *classes
    *attributes
    *relations
    *object oriented diagram
  dynamic design
    *interactions
    *interaction diagram
    *scenarios
java applet
  *applet methods
  HTML tag
   ..
user interface
  event model
    *event source
```

```
    *event listener
    adapter
  *events
     action event
     text event
     item event
     adjustment event
  *AWT classes
     panel
     frame
     ..
..
```

In the conceptual model of the hyperbook (figure 4.8), we group the portfolio concepts of a student group according to the area concepts. As an area consists of several knowledge items, we are able to integrate the portfolios of the students by using the index of each of the portfolio concepts. Thus, we define both the basic structure of student projects as well as their connection to the remainder of the course material.

Figure 4.10 shows the portfolio of a student group called "Bleifuss" which has implemented a car racing game in Java.

Clearly, an important part of a portfolio is also the self presentation of the students who have worked it out. Therefore the student's homepages are also integrated in the hyperbook (see relation between the concepts *student group* and *portfolio* in figure 4.8). Figure 4.11 shows as an example the homepage of the student group "Spacemen" which worked out a portfolio in a CS1 course.

4.3 System Architecture

The KBS hyperbook system is implemented entirely in Java. A servlet residing in the Java Web Server (see fig. 4.12) 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 (see section 7.2), are also realized by Java client applets.

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

Storage Module

Base 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

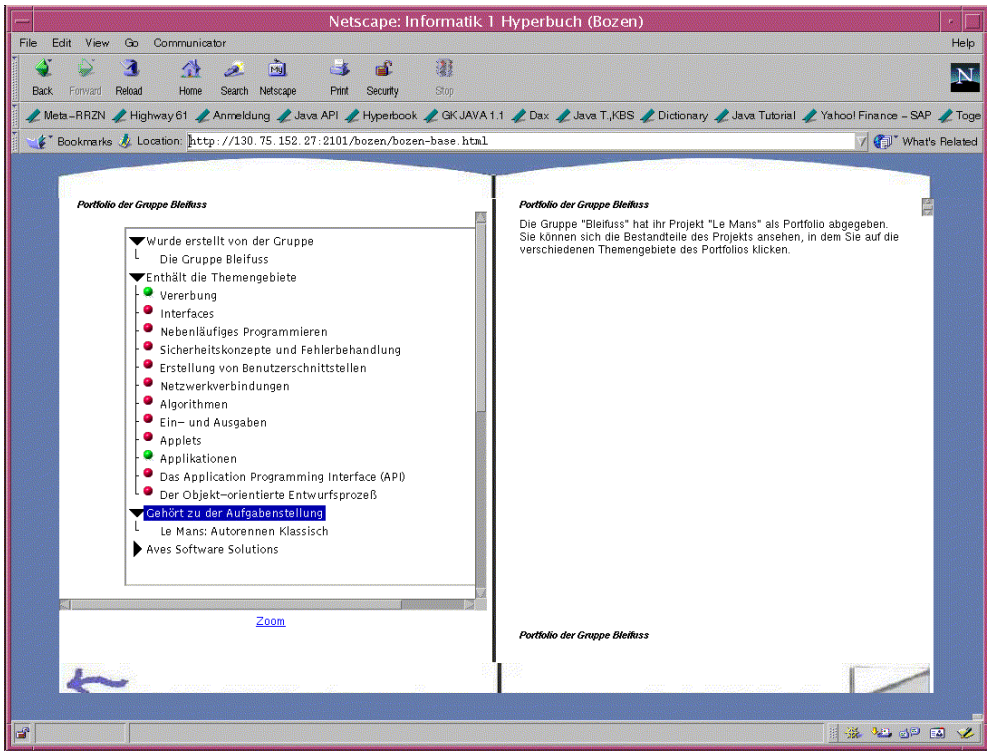

Figure 4.10: Portfolio of the student group "Bleifuss"

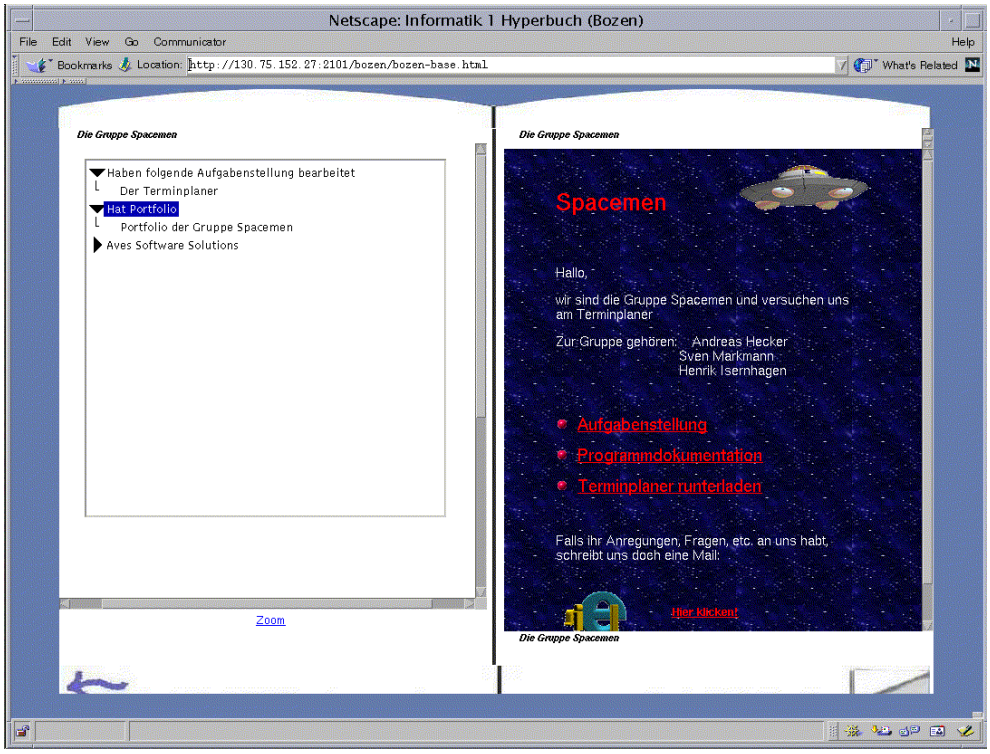

Figure 4.11: Homepage of the student group "Spacemen".

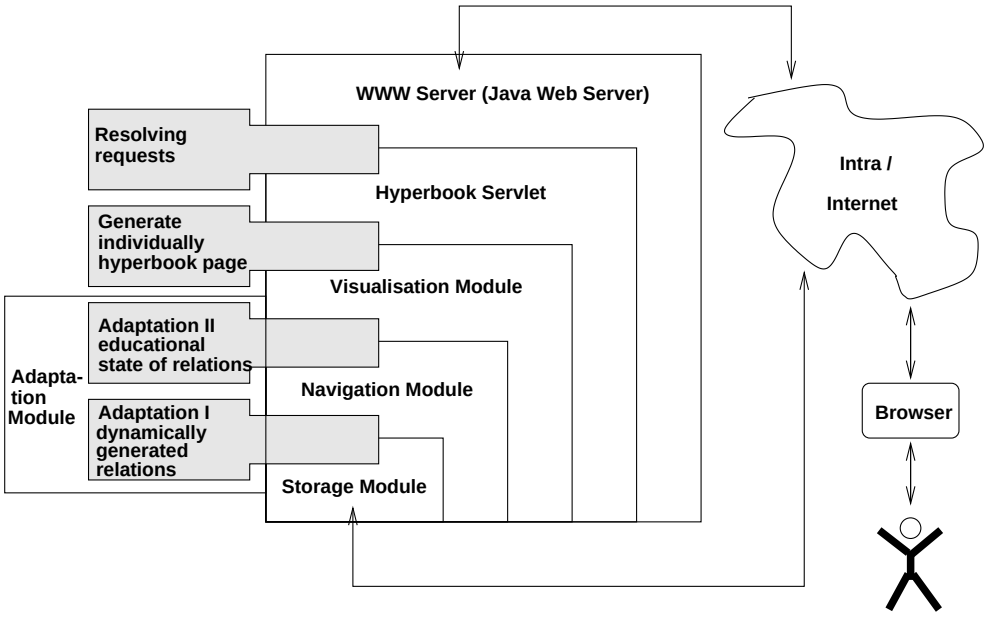

Figure 4.12: Schematic view of the implementation of the hyperbook system

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 4.1), 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 (see section 7.2.3), are returned. For the adaptation of instances of the course concept, a next reasonable learning goal according to the user's knowledge is generated (section 7.3), and the navigation module receives a corresponding relation. If a student defines a learning goal on himself or requests a next learning goal from the hyperbook system, text units containing useful information for this particular project

are selected (section 7.2). In addition, a learning sequence containing information necessary for reaching the learning goal is generated (section 7.5), and projects are selected from the project library (section 7.5).

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 dynamically 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.

Chapter 5

Adaptation Component of the KBS Hyperbook System

One of the main goals of student modeling in educational hypermedia is student guidance [6]. Students have learning goals and previous knowledge which should be reflected by the hyperbook for adapting the content or the link structure of the hyperdocument. For our KBS hyperbook system we follow a constructivist pedagogic approach, building on project based learning, group work, and discussions (see chapter 3) [48]. 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 current student 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.

In this chapter we will describe the concept and realization of the *adaptation component* for KBS hyperbooks.

This student modeling component 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 origin from the hyperbook's libraries or from any other location in the WWW.

5.1 Modeling the Knowledge Domain

5.1.1 Knowledge Items

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*($\mathcal{KI}$). Knowledge items are similar to the domain model concepts used in [8] or the knowledge units in [22].

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

We put a partial order on the set of $\mathcal{KI}$s to represent learning dependencies. $KI_1 < KI_2$ 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 $\mathcal{KI}$ `control_structures`, it is necessary to know about the $\mathcal{KI}$s `branching` and `looping`, thus

`looping` < `control_structures` and `branching` < `control_structures`.

5.1.2 Describing the Knowledge Domain

To identify relevant knowledge items of the application domain of a hyperbook, we first specify the main knowledge items of the book. This can be done in different ways, depending on the application domain. For example, the hyperbook for our course "Introduction to Java Programming" contains the following main $\mathcal{KI}$s, which we have related to the ACM Computing Classification System (1998 version)[1]:

- Language Constructs and Features (D.3.3)
  1. `classes_and_objects` (D.3.3)
  2. `methods` (D.3.3 – Procedures, functions and subroutines)
  3. `inheritance` (D.3.3)
  4. `interfaces` (D.3.3 – Polymorphism resp. D.2.2 Design Tools and Techniques – Modules and Interfaces)
  5. `data_types_and_operators` (D.3.3 – Data types and structures resp. subtopics from E.1 Data structures)
  6. `control_structures` (D.3.3)
  7. `concurrent_programming_structures` (D.3.3)
  8. `error_handling_and_recovery` (D.2.5 Testing and Debugging – Error handling and recovery)[2]
  9. `java_windowing_system` (H.5.2 User Interfaces – Windowing Systems)
  10. `java_networking_constructs` (C.2.5 Local and Wide-Area Networks – Internet)
  11. `algorithms` (D.3.3. Recursion resp. E.5 Files – Searching/Sorting)
  12. `input_output` (D.3.3)

[1] http://www.acm.org/class/1998/ccs98.html

[2] would better be classified in D.3.3

13. `java_applets` (related to D.3.3 and D.3.4 Processors – Run-time environment)
14. `java_applications` (related to D.3.3 – Procedures, functions and subroutines)[3]

- Additional Java specific topics
  15. `java_api` (D.2.2 Software Libraries and D.3.3 Modules, packages)
  16. `programming_environment` (subtopics from D.2.3 Coding Tools and D.3.4 Processors)
- Introductory software engineering concepts
  17. `object_oriented_design_methods` (D.2.2)

Each of these $\mathcal{KI}$s is accompanied with a set of knowledge items describing the aspects of a topic more precisely. For example, `control_structures` has the subtopics `branching`, `looping`, `return_statement`, `exit_statement`, `execution_block`. The $\mathcal{KI}$ `branching` itself has the subtopics `if` and `switch`.

The resulting set of 289 knowledge items can be seen in appendix A.

5.2 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 $\mathcal{U}$ has knowledge about a topic KI, on the base of all observations $\mathcal{E}$ the system has about $\mathcal{U}$:

Definition 5 (Knowledge Vector) $\vec{KV}$ *:*

$$\vec{KV}(\mathcal{U}) = (P(KI_1|\mathcal{E}), P(KI_2|\mathcal{E}), \ldots, P(KI_n|\mathcal{E}))\,,$$

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

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

A student can have

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

[3] thematically related to Methods, Classes and objects, and to Data types and operators

Thus, the $\mathcal{KI}$ s 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 the 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.

5.3 Indexing Information: HTML pages, Examples, Projects

In this section, we will see how to index various information resources.

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

Definition 6 (Content Map) *Let $\mathcal{S} \neq \emptyset$ be the set of all $\mathcal{KI}$ s, and let $\mathcal{H}$ be a set of HTML pages. Then*

$$I : \mathcal{H} \rightarrow \mathcal{P}(\mathcal{S}) \backslash \{\emptyset\} \tag{5.1}$$

is the content map, which gives for each information resource in $\mathcal{H}$ the index of this resource, e.g. the set of $\mathcal{KI}$ s describing its content.

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

5.3.1 Indexing HTML pages

An HTML page is indexed by those $\mathcal{KI}$ s 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 $\mathcal{KI}$ 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, as can be seen in section 7.2.2.

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 $\mathcal{KI}$: The* `what_is-`$\mathcal{KI}$ *of the* $\mathcal{KI}$ `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 $\mathcal{KI}$ 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* $\mathcal{KI}$ `public_static_void_main`. *If this page also contains information about reading command line parameters, the index will contain the* $\mathcal{KI}$ *s* `public_static_void_main` *and* `input_parameter`.

The hierarchy of the knowledge items can be used for abbreviating the indexing process.

Example 4 *If a page H contains basic information about applications then its index* $I(H)$ *may consist of* `system_out_println`, `input_parameter`, `public_static_void_main`, `executable`. *As can be seen in the complete list of all* $\mathcal{KI}$ *s (appendix A), the above mentioned* $\mathcal{KI}$ *s are exactly the subtopics of the* $\mathcal{KI}$ `java_application`. *To avoid unnecessary work for indexing, the author can also use the* $\mathcal{KI}$ `java_application` *for indexing the content of this particular page:* $I(H) = \{$`java_application`$\}$.

5.3.2 Indexing Chapters, Areas, etc.

In the KBS hyperbook system we are using the concept of areas to partition the application domain (see section 4.2). For example, the concepts of the ACM classification (see section 5.1.2) are used as area concepts for the CS1 hyperbook. These area pages would need two different index sets. One for describing their range as an area. And another one for indexing their content. To simplify the indexing step and to avoid the use of two different index sets, we ask the authors of such pages to index them with all relevant $\mathcal{KI}$ s of this chapter or area, while the first item in the index is the `what_is-`$\mathcal{KI}$ describing the content of the actual page.

5.3.3 Indexing Portfolios

A portfolio concept is indexed by the $\mathcal{KI}$ which describe the particular content of this portfolio page (see section 4.2.4).

5.3.4 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 $\mathcal{KI}$ s,

higher level $\mathcal{KI}$s can be used as well. To emphasize specific topics as *very important* or *very good explained* in an example, the author of a project can provide weights with each $\mathcal{KI}$ indexing the project. Such a weight is the percentage of a $\mathcal{KI}$ in this particular project.

Example 5 *The "threads in applets" project shows the use of threads for loading data in applets. It is indexed by the $\mathcal{KI}$s* `java_applet` *(30 %) and* `single_thread` *(70 %).*

5.3.5 Update: Observations about Users

Several systems detect the fact that the student reads some information to update the estimate of his knowledge (e.g. [8]). 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 [6]. 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, see section 3.5. For motivating the students to explore the learning material by doing some projects, the hyperbook is only updated whenever 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 $\mathcal{KI}$ (see section 5.2). The second way is to ask some experts for judging the student's project performance.

5.3.6 Indexing Constraints

For obtaining a correct indexing of the hyperbook (see section 5.3), the index set has to obey the following constraint, which is motivated by the project-based learning approach (see section 3) underlying our hyperbooks. As we have seen in section 5.3.5, 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 $\mathcal{KI}$ must be contained in an index set of some project:

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

5.4 Discussion

In section 5.3 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 new. Most

approaches model dependencies like prerequisites or outcomes directly with the information resources themselves.

We separate knowledge and information, as we model learning dependencies solely on the set of $\mathcal{KI}$s of a hyperbook. The connection between the student modeling component and the hyperbook system is the content map (see definition 6), which maps each information resource to a set of $\mathcal{KI}$s.

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 to include 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, this openness is enabled by the indexing concept, too. In addition, as we will see in chapter 7, 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 it would be necessary if knowledge, and thus reading or learning dependencies, would have been coded in the material itself.

The chosen way of 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 (see chapter 6).

The adaptation component has the following features:

- robust against changes,
- different information origins (hyperbook, electronic books, arbitrary information resources in the WWW),
- multiple authors of a hyperbook, which may work independently,
- only one student modeling component serving for an arbitrary dependency engine.

Chapter 6

Bayesian Network Engine: Calculating Probabilities

So far, we have seen how to use knowledge items for describing the knowledge domain, for describing a student's knowledge, and for indexing different kinds of information resources.

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 (section 5.2).

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 $\mathcal{KI}$, thus for calculating the knowledge vector of our users.

Definition 7 (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 of 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 $\mathcal{KI}$ 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 (see section 6.1). Second, defining probability tables for all vertices (see section 6.2).

6.1 The Bayesian Network Graph

6.1.1 Dependencies of the $\mathcal{KI}$

With the partial order defined in section 5.1, we construct some dependency graph for the $\mathcal{KI}$s of a hyperbook. Therefore we first look at the *main* $\mathcal{KI}$s of the hyperbook which describe the areas of the application domain. In the case of the CS1 hyperbook, these are the $\mathcal{KI}$s of the ACM classification (section 5.1.2).

The dependency graph is the neighbouring graph of the partial ordering $<$ of the set of $\mathcal{KI}$s, i.e. there is an edge from X to Y if $Y < X$ and there exists no Z with $Y < Z < X$. The subgraph of the dependency graph of the CS1 hyperbook which contains only the main $\mathcal{KI}$s, can be seen in figure 6.1.

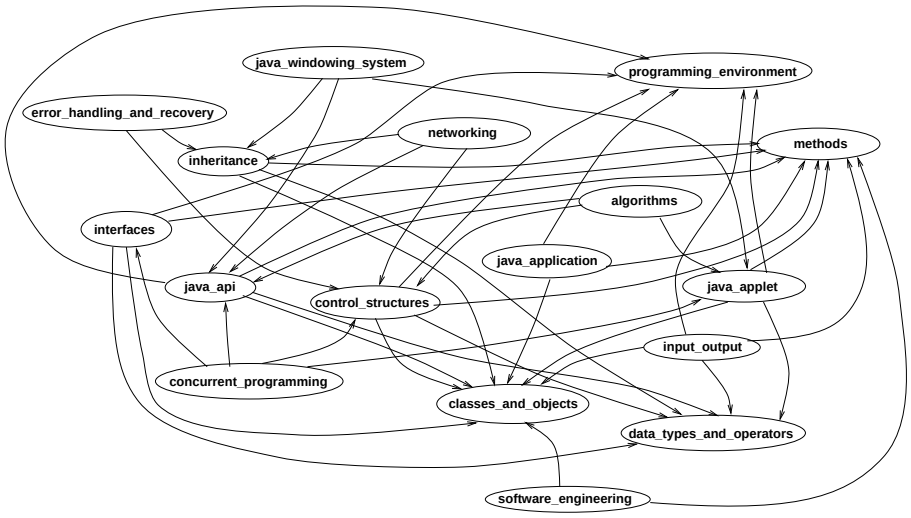

Figure 6.1: Dependency graph of the $\mathcal{KI}$s of the ACM classification

As each $\mathcal{KI}$ from the ACM classification has a set of $\mathcal{KI}$s describing it in more detail (section 5.1.2), we construct for each of those main $\mathcal{KI}$s the dependency graph containing the $\mathcal{KI}$ together with its subconcepts. In the sequel we assume that these graphs are rooted trees, having the main concept as its root. The root tree containing the $\mathcal{KI}$ `classes_and_objects` and its subtopics is given in figure 6.2.

The dependency graph of the $\mathcal{KI}$s of the ACM classification together with the rooted trees of each $\mathcal{KI}$ is the *basic graph* for our Bayesian network.

To further structure this basic graph, we categorize the main $\mathcal{KI}$s into three level according to the following aspects: The *first level* contains the simple topics, which require no prerequisite knowledge for understanding. The *second level* contains advanced topics which require knowledge about some of the level 1 concepts. Finally, the *third level* is the level of the compound topics which can only be understood by knowing some first- or second-level topics. We assume in the sequel that distinct main $\mathcal{KI}$s within the same level are not related, i.e. independent random variables. The second level is further divided into two parts: one that is necessary to understand some of the third-level concepts (level 2 in figure 6.3) and another part that is not required by any third-level concept (level 2' in figure 6.3). Optional topics can

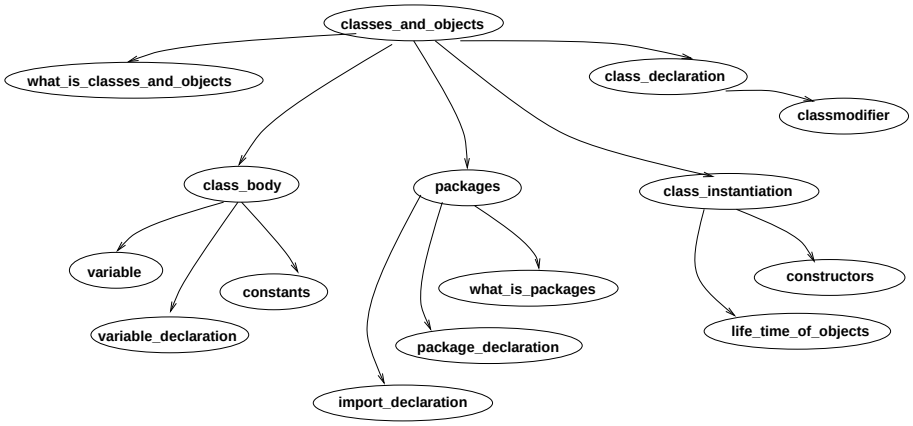

Figure 6.2: Rooted tree of the ACM classification $\mathcal{KI}$ `classes_and_objects`

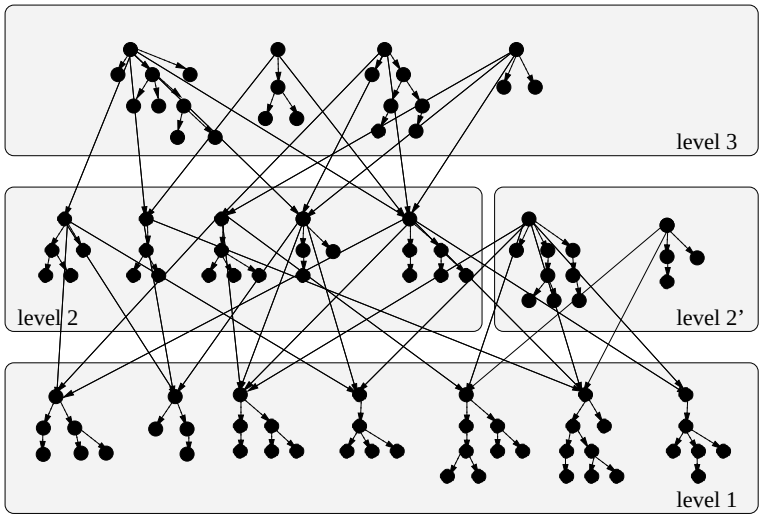

Figure 6.3: Schematic model of a Bayesian network underlying the user model

only occur in level 2' and level 3. If we want to use optional topics, these topics are not contained in a user's knowledge vector as the user can also read the hyperbook by skipping them.

For the CS1 hyperbook, the sectioning into level is the following:

- Level 1 consists of the topics `classes_and_objects`, `methods`, `data_types_and_operators`, `programming_environment`, and `object_oriented_concepts_in_software_engineering`.
- Level 2' consists of the topics `input_output`, `java_application`, and `object_oriented_design_methods`.
- Level 2 contains `control_structures`, `interfaces`, `inheritance`, `java_applet`, `java_api`.

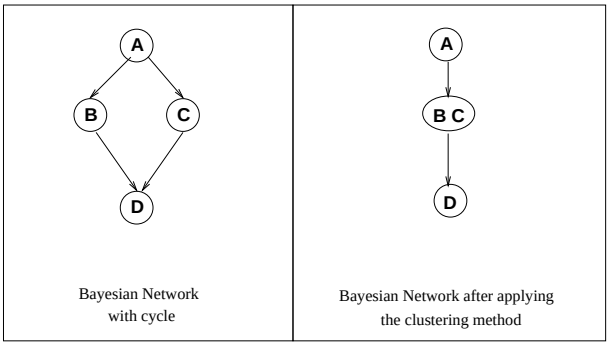

Figure 6.4: Example for eliminating a not continuously directed cycle with clustering

- Level 3 `java_windowing_system`, `error_handling_and_recovery`, `concurrent_programming_structures`, `java_networking_constructs`, `algorithms`.

6.1.2 YACF: Yet Another Clustering Formalism

For our concerns, we construct a Bayesian network with random variables which give a probability distribution for calculating the knowledge of a user.

As we have already seen in section 5.2, we are using random variables with four discrete values. Recall that they have all the same range {**E, F, A, N**} of size 4. Thus, we assume that all random variables have the same number of discrete values. This is only a matter of simplification. Of course we can also have random variables with different values; the following considerations and theorems remain true.

Exact inference in Bayesian networks is known to be NP-hard [16]. A general Bayesian network can represent any propositional logic problem (if all probabilities are 1 or 0), and propositional logic problems are known to be NP-hard [101].

Linear time algorithms exist for Bayesian networks which have no cycles in the underlying undirected graph. There are several methods how to deal with such not continuously directed cycles: clustering, conditioning and stochastic simulation algorithms.

Clustering algorithms glue two or more nodes together to avoid not continuously directed cycles (see for example [76, 21]). An example for a clustering method can be seen in figure 6.4. Here, the variables B and C are clustered to one single node BC. The range of BC is the cross product of the ranges of B and of C. To query B or C, one has to average over the values of the other variable in the cluster node.

Finding the right cluster nodes is the important part of clustering. Since the conditional probability tables for the cluster nodes contain the cross product of the range of their clustered variables, the number of required calculations for querying the network may increase exponentially [101].

In case of our previously constructed BN, we have a graph containing lots of not continuously directed cycles, as can be seen in figure 6.3. Attempts to apply clustering to this BN led to networks with a large number of cluster nodes. It was necessary to

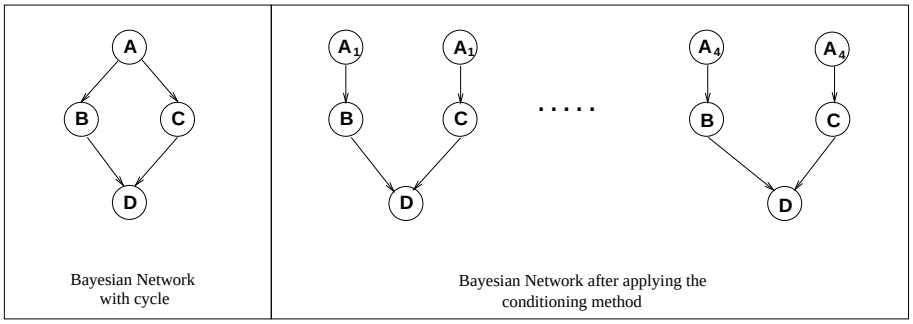

Figure 6.5: Example for eliminating a not continuously directed cycle with conditioning

cluster nodes more than twice, thus querying the $\mathcal{KI}$ s in the so constructed network became also dependent on the number of average calculations.

Conditioning methods transform the network into several simpler networks (see for example [60, 20]). Each of these networks contains one or more of the random variables instantiated to one of their values. For example, in figure 6.5, A is a random variable with four discrete values, and the conditioning method leads to four new networks, in which the variable A is instantiated to one of it's four values.

As the number of networks in the conditioning method increases exponentially with the number of instantiated variables and their values, this approach is disadvantageous for our application, as we have to propagate a large number of so constructed Bayesian networks.

Stochastic simulation methods run repeatedly simulations of the network for calculating approximations of the exact evaluation (see for example [106, 43]. Even with using likelihood weighting [35, 106], the performance is problematic. In the CPSC project [95], for example, a BN with 448 nodes and 906 links needs around 35 minutes for calculating accurate values. The network of our CS1 hyperbook contains 289 nodes, so answers in realtime were not possible with this approach. Hence we decided to investigate on a solution tailored to our application domain.

For a discussion on the advantages of probabilistic reasoning compared to fuzzy logic methods or rule-based reasoning in artificial intelligence see [92, 109, 93, 107].

We developed a special clustering formalism which enables us to generate a directed graph without cycles in the underlying undirected graph (see section 6.1.2). The algorithm we are then able to apply for Bayesian inference in this generated graph is an exact inferring algorithm [101]. We have reimplemented this algorithm in Java. It needs in average 1/5 seconds on a Sparc 10 machine to query a node.

The YACF clustering formalism introduces a new, additional cluster node while nearly all other nodes remain unchanged, only the conditional probability tables of the child vertices of the cluster had to be modified.

This additional cluster node takes the information of the parent nodes and passes it to the child nodes, as can be seen in figure 6.6. Thereby, the not continuously directed cycles in the dependencies between parents and children are deleted. The cluster node distribute the evidence from its parents directly to its children while

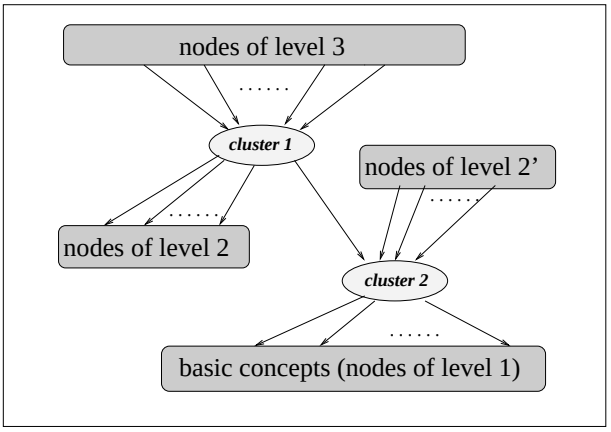

Figure 6.6: Graph after applying the YACF clustering twice

evidence in child nodes which is propagated to the parent nodes is very poor. This is the kind of updating we need for our BN. For example, the observation that a student masters the $\mathcal{KI}$ `control_structures` does not say anything secure about his knowledge about `algorithms` or `event_model`. But if we see the student implementing a search algorithm, we can say with high probability that he knows how to control the program flow. Therefore we are mainly interested in carrying information from parents to child nodes.

To construct a YACF cluster which distributes information from parents to children (see figure 6.7), we define this node as a random variable whose range is the sum of the ranges of all child nodes. By doing this, we have exactly one part of the range of the cluster variable which holds for a particular child node. Hence each child has to listen only to that part of the cluster variable's range which holds information for it. For example, we observe a user knowing a $\mathcal{KI}$ X_1 with grade expert's knowledge (E) and a $\mathcal{KI}$ X_2 with grade beginner's knowledge (A), and both X_1, X_2 are parents of a $\mathcal{KI}$ Y. The information we pass to the child Y is then the best grade of knowledge the user has in all parent variables. In the example, the grade of knowledge we distribute to Y is E.

The following definitions show, how we construct the conditional probability tables for such a cluster node and its children. Theorem 6.1 shows that the children of a cluster vertex depend on their original parents, thus with the YACF clustering we found a formalism which serves for our purposes.

Definition 8 (Cond. Probability Table (CPT) for a YACF-Cluster Node) *Let $X_1, \ldots, X_N$, $Y_1, \ldots, Y_M$ random variables. Each Y_i, $i \in \{1, \ldots, M\}$ is dependent from at least one $X_k, k \in \{1, \ldots, M\}$. The conditional probability table of the cluster node H is a $\prod_{l=1}^{N} |R(X_l)| \times \sum_{i=1}^{M} |R(Y_i)|$ matrix, where $R(X)$ denotes the range of a discrete random variable X. Let $1Y_i, \ldots, LY_i$ denote the part of the range of H which carries information for node Y_i. Then*

$$P\left(H = kY_i | (X_1, \ldots, X_N) = (x_1, \ldots, x_n)\right) =$$
$$\begin{cases} \frac{1}{M} & , \textit{ if } k = \textbf{best\_grade}\left(x_l | l \in \{1, \ldots, n\} \wedge Y, X_l \textit{ dependent }\right) \\ 0 & , \textit{ else} \end{cases}$$

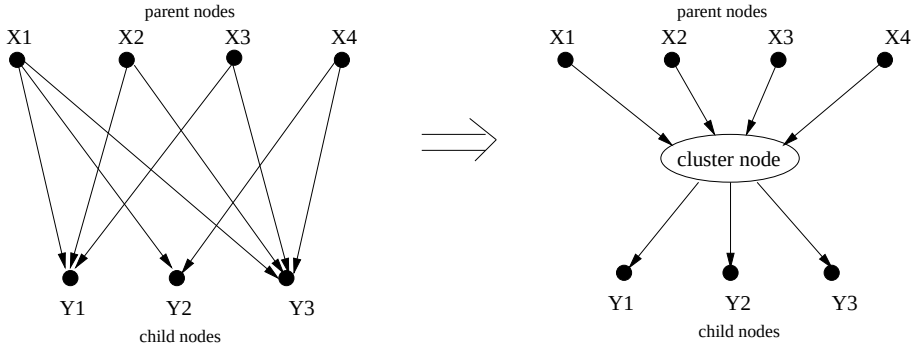

Figure 6.7: The role of a YACF cluster node

	range of H which holds evidence for node Y1					range of H which holds evidence for node YM			
X1 ... XK .... XN	P(H=E_Y1|..)	...	P(H=N_Y1|..)			P(H=E_YM|..)		P(H=N_YM|..)	
X1=E, ... XK=E, ... XN=E	1/M	0	0	0	.....	1/M	0	0	0
X1=F, ... XK=N, ... XN=E	0	1/M	0	0	.....	1/M	0	0	0
X1=N, ... XK=E, ... XN=N	1/M	0	0	0	.....	0	0	0	1/M

Figure 6.8: CPT of a YACF-Cluster node with N parents and M children.

where best_grade returns a maximum value of the random variables, e.g. in our interpretation, the best grade of knowledge we observe in all the parent variables Y depends on.

Note: the random variable of the cluster node has as many values as the sum of all values of the children of the cluster. Figure 6.8 gives an example for a CPT of a cluster node.

The probability table assigned to a child node of a cluster node consists of two parts: a "blueprint" table for those values of the random variable "cluster node" which correspond to this particular child node, and an equal distribution for all other values. An example of such a CPT for four-value discrete random variables can be see in figure 6.9.

Definition 9 (CPT for child nodes of a YACF Cluster Node) *Let $X_1, \ldots, X_N, Y$ be random variables, Y is dependent from at least one $X_k, k \in \{1, \ldots, M\}$. Let $R(Y)$ be the range of Y and let $\{1Y, \ldots, LY\}$ denote the part of the range of H that holds the information for Y. Then the CPT of a YACF cluster node H with parents $X_1, \ldots, X_N$ and child Y is defined as follows:*

$$P(Y|H=h) \quad = \quad \begin{cases} \left(\frac{1}{|R(Y)|}, \ldots, \frac{1}{|R(Y)|}\right) & , \textit{if } h \notin \{1Y, \ldots, LY\} \\ P(Y|X=h) & , \textit{else} \end{cases}$$

Cluster Node (H)	P(Y=E | H=...)	P(Y=F | H =...)	P(Y=A | H=...)	P(Y=N | H = ...)	
H_1 = E	0.25	0.25	0.25	0.25	
	⋮			⋮	
H_{i-1} = N	0.25	...	...	0.25	
H_i = E	0.8	0.2	0	0	range of H
H_i = A	0.2	0.6	0.2	0	holding
H_i = F	0	0.2	0.6	0.2	evidence
H_i = N	0	0	0.2	0.8	for Y
H_{i+1} = E	0.25	0.25	0.25	0.25	
	⋮			⋮	
H_n = N	0.25	...	...	0.25	

Figure 6.9: Conditional probability table for child node Y of cluster node H

Theorem 1 (Cluster nodes serve as probability distributors) *Let $X_1, \ldots, X_N$ the parents of a cluster node H and $Y_1, \ldots, Y_M$ the children of H. Let $I_j := \{i \in \{1, \ldots, M\} : P(Y_i|X_j) \neq P(Y_i)\}, j \in \{1, \ldots, N\}$.*

Then

$$P\left(Y|\bigwedge_{i=1}^{N} X_i\right) \stackrel{!}{=} P\left(Y|\bigwedge_{\substack{i=1\\ i\in I(Y)}}^{N} X_i\right) \tag{6.1}$$

where

$$I(Y) = \{l : Y, X_l \text{ are dependent}\}$$

Proof: Let WH be the range of H and let $\{1Y_i, \ldots, LY_i\}$ denote the part of the range of H that holds the evidence for Y_i.

Let $\vec{WX}_{i,i\notin I(Y)}$ the be range of those X_i, from which Y is conditional independent:

$$\vec{WX}_{i,i\notin I(Y)} := W\vec{X}_{i_1} \times \ldots \times W\vec{X}_{i_k}, \qquad i_l \in \{1, \ldots, M\}\backslash I(Y)$$

First we handle the left side of the stated equation:

$$\begin{aligned}
P\left(Y|\bigwedge_{i=1}^{N} X_i\right) &= \sum_{h\in WH} P\left(Y, \bigwedge_{i=1}^{N} X_i, H=h\right) \cdot \frac{1}{P\left(\bigwedge_{i=1}^{N} X_i\right)} \\
&= \sum_{h\in WH} P\left(Y|\bigwedge_{i=1}^{N} X_i, H=h\right) \cdot P\left(H=h|\bigwedge_{i=1}^{N} X_i\right) \\
&\stackrel{\substack{\text{cond.}\\ \text{independent}}}{=} \sum_{h\in WH} P(Y|H=h) \cdot P\left(H=h|\bigwedge_{i=1}^{N} X_i\right) \\
&= \sum_{h=1Y_1}^{LY_1} P(Y|H=h) \cdot P\left(H=h|\bigwedge_{i=1}^{N} X_i\right)
\end{aligned}$$

$$+ \sum_{h=1Y_2}^{LY_2} P(Y|H=h) \cdot P\left(H=h|\bigwedge_{i=1}^{N} X_i\right) + \dots$$
$$+ \sum_{h=1Y_M}^{LY_M} P(Y|H=h) \cdot P\left(H=h|\bigwedge_{i=1}^{N} X_i\right)$$

If $Y_i \neq Y$, we obtain

$$\sum_{h=1Y_i}^{LY_i} \underbrace{P(Y|H=h)}_{=\frac{1}{L}\forall h} \quad \underbrace{P\left(H=h|\bigwedge_{i=1}^{N} X_i\right)}_{\substack{=\frac{1}{N} \text{ for exactly one } h(Y)\in\{1Y_i,\dots,LY_i\},\\ =0 \text{ in all other cases}}}$$
$$= \frac{1}{L \cdot N} \tag{6.2}$$

In case of $Y_i = Y$, we have

$$\sum_{h=1Y}^{LY} P(Y|H=h) \quad \underbrace{P\left(H=h|\bigwedge_{i=1}^{N} X_i\right)}_{\substack{=\frac{1}{N} \text{ for exactly one } h(Y)\in\{1Y,\dots,LY\},\\ =0 \text{ in all other cases}}}$$
$$= \frac{1}{N} \cdot P(Y|H=h(Y)) \tag{6.3}$$

With 6.2 and 6.3 the left hand side of equation 6.1 yields to:

$$P\left(Y|\bigwedge_{i=1}^{N} X_i\right) = (M-1) \cdot \frac{1}{L \cdot N} + \frac{1}{N} \cdot P(Y|H=h(Y)) \tag{6.4}$$

Now we handle the right hand side of equation 6.1:

$$P\left(Y|\bigwedge_{\substack{i=1\\ i\in I(Y)}}^{N} X_i\right) = \sum_{h\in WH} P(Y|H=h) \cdot P\left(H=h|\bigwedge_{\substack{i=1\\ i\in I(Y)}}^{N} X_i\right)$$
$$= \sum_{h\in WH} P(Y|H=h)$$
$$\cdot \sum_{\substack{\vec{w}\in W\vec{X}_i\\ i\notin I(Y)}} P\left(H=h|\bigwedge_{\substack{i=1\\ i\in I(Y)}}^{N} X_i \wedge \bigwedge_{\substack{i=1\\ i\notin I(Y)}}^{N} X_i = \vec{w}\right)$$
$$\cdot \underbrace{P\left(\bigwedge_{\substack{i=1\\ i\notin I(Y)}}^{N} X_i = \vec{w}|\bigwedge_{\substack{i=1\\ i\in I(Y)}}^{N} X_i\right)}_{=\frac{1}{L^{N-I}}}$$

$$
\begin{aligned}
= & \sum_{h=1Y1}^{LY1} P\left(Y|H=h\right) \cdot \sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} P\left(H=h|\bigwedge_{\substack{i=1 \\ i\in I(Y)}}^{N} X_i \wedge \bigwedge_{\substack{i=1 \\ i\notin I(Y)}}^{N} X_i=\vec{w}\right)\cdot\frac{1}{L^{N-I}} \\
& + \ldots \\
& + \sum_{h=1YM}^{LYM} P\left(Y|H=h\right) \cdot \sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} P\left(H=h|\bigwedge_{\substack{i=1 \\ i\in I(Y)}}^{N} X_i \wedge \bigwedge_{\substack{i=1 \\ i\notin I(Y)}}^{N} X_i=\vec{w}\right)\cdot\frac{1}{L^{N-I}}
\end{aligned}
$$

In case of $Y_i \neq Y$, we have

$$
\begin{aligned}
& \sum_{h=1Yi}^{LYi} \underbrace{P\left(Y|H=h\right)}_{=\frac{1}{L}\forall h} \cdot \sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} P\left(H=h|\bigwedge_{\substack{i=1 \\ i\in I(Y)}}^{N} X_i \wedge \bigwedge_{\substack{i=1 \\ i\notin I(Y)}}^{N} X_i=\vec{w}\right)\cdot\frac{1}{L^{N-I}} \\
\overset{\sum \text{changed}}{=} & \sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} \sum_{h=1Yi}^{LYi} \frac{1}{L} \cdot \underbrace{P\left(H=h|\bigwedge_{\substack{i=1 \\ i\in I(Y)}}^{N} X_i \wedge \bigwedge_{\substack{i=1 \\ i\notin I(Y)}}^{N} X_i=\vec{w}\right)}_{\substack{=\frac{1}{N} \text{ for exactly one } h(Y_i)\in\{1Y_i,\ldots,LY_i\}, \\ =0 \text{ in all other cases}}} \cdot\frac{1}{L^{N-I}} \\
= & \sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} \frac{1}{L}\cdot\frac{1}{N}\cdot\frac{1}{L^{N-I}} \\
= & \frac{1}{L\cdot N}\cdot\frac{1}{L^{N-I}}\cdot \underbrace{\sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} 1}_{\substack{\vec{w} \text{ has } N-I \text{ components}, \\ L \text{ possible values per component}\rightarrow L^{N-I} \text{ terms}}} \\
= & \frac{1}{L\cdot N}\cdot\frac{1}{L^{N-I}}\cdot L^{N-I} \\
= & \frac{1}{L\cdot N}
\end{aligned}
\tag{6.5}
$$

In case of $Y_i = Y$, we have

$$
\begin{aligned}
& \sum_{h=1Y}^{LY} P\left(Y|H=h\right) \cdot \sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} P\left(H=h|\bigwedge_{\substack{i=1 \\ i\in I(Y)}}^{N} X_i \wedge \bigwedge_{\substack{i=1 \\ i\notin I(Y)}}^{N} X_i=\vec{w}\right)\cdot\frac{1}{L^{N-I}} \\
\overset{\sum \text{changed}}{=} & \sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} \sum_{h=1Y}^{LY} P\left(Y|H=h\right)\cdot \\
& \qquad P\left(H=h|\bigwedge_{\substack{i=1 \\ i\in I(Y)}}^{N} X_i \wedge \bigwedge_{\substack{i=1 \\ i\notin I(Y)}}^{N} X_i=\vec{w}\right)\cdot\frac{1}{L^{N-I}} \\
= & \sum_{\substack{\vec{w}\in W\vec{X}_i \\ i\notin I}} \sum_{h=1Y}^{LY} P\left(Y|H=h\right)\cdot
\end{aligned}
$$

$$\underbrace{P\left(H = h|\bigwedge_{\substack{i=1\\ i\in I(Y)}}^{N} X_i\right)}_{\substack{=\frac{1}{N} \text{ for exactly one } h(Y)\in\{1Y,\ldots,LY\}\\ =0 \text{ in all other cases}}} \cdot\frac{1}{L^{N-I}}$$

$$= \sum_{\substack{\vec{w}\in W\vec{X}_i\\ i\notin I}} P\left(Y|H = h(Y)\right)\cdot\frac{1}{N}\cdot\frac{1}{L^{N-I}}$$

$$= \frac{1}{N}\cdot\frac{1}{L^{N-I}}\cdot P\left(Y|H = h(Y)\right)\cdot\underbrace{\sum_{\substack{\vec{w}\in W\vec{X}_i\\ i\notin I}} 1}_{=L^{N-I}}$$

$$= \frac{1}{N}\cdot P\left(Y|H = h(Y)\right) \tag{6.6}$$

With 6.5 and 6.6 we obtain for the right hand side of the stated equation:

$$P\left(Y|\bigwedge_{\substack{i=1\\ i\in I(Y)}}^{N} X_i\right) = (M-1)\cdot\frac{1}{L\cdot N} + \frac{1}{N}\cdot P\left(Y|H = h(Y)\right) \tag{6.7}$$

If we compare 6.4 and 6.7, we obtain the stated equation. q.e.d.

Corollary 1 *Let $X_i, Y_j, i \in \{1,\ldots,N\}, j \in \{1,\ldots,M\}$ be two independent random variables of the BN. Let Y_j be a child node of X_i via the cluster node $\mathcal{H}$ of theorem 6.1. Then*

$$P(Y_j|X_i) = P(Y_j)$$

Proof: With Theorem 6.1 we obtain:

$$\begin{aligned}
P\left(Y_j|X_i\right) &\overset{\text{total prob. formula}}{=} \sum_{\vec{X}=X_1\times\ldots X_{i-1}\times X_{i+1}\times\ldots\times X_N} P\left(Y_j|X_i\wedge\vec{X}=\vec{x}\right)\cdot P\left(\vec{X}=\vec{x}|X_i\right)\\
&\overset{\text{Theorem 6.1}}{=} \sum_{\vec{X}=X_1\times\ldots X_{i-1}\times X_{i+1}\times\ldots\times X_N} P\left(Y_j|\vec{X}=\vec{x}\right)\cdot P(\vec{X}=\vec{x})\\
&\overset{\text{cond. prob.}}{=} \sum_{\vec{X}=X_1\times\ldots X_{i-1}\times X_{i+1}\times\ldots\times X_N} P\left(Y_j\wedge\vec{X}=\vec{x}\right)\\
&\overset{\text{total prob. formula}}{=} P(Y_i).
\end{aligned}$$

This shows that $P(A|Y)$ is the sum of all conditional probabilities of those nodes that influence A. If no X_i carries new evidence (that is the case when all the evidence propagated in the BN is Y), the value of $P(A|Y)$ equals the old value $P(A)$.

If we apply the YACF clustering formalism twice, we obtain a directed graph without cycles in the underlying directed graph (see figure 6.6). Here, we used a YACF cluster node for eliminating not continuously directed cycles between level 3 and 2 concepts and another one for eliminating not continuously directed cycles between level 2 and level 1 concepts. Thus we are ready with step one of the construction of our Bayesian network.

6.2 Finding Conditional Probability Tables

After generating the directed acyclic graph out of the dependency graph, we have to add for each node of the graph a probability table containing the conditional probabilities a child node has with respect to his parent node.

sorting	P(quicksort=E|.)	P(quicksort=F|.)	P{quicksort=A|.)	P{quicksort=N|.)
E	0.8	0.2	0	0
F	0.2	0.6	0.2	0
A	0	0.2	0.6	0.2
N	0	0	0.2	0.8

Figure 6.10: conditional probability table for the node `quicksort`, which is dependent on the node `sorting`

We provide some "blueprint" tables that express how strong a dependency between a parent node and a child node is. We assume for example, that a user familiar with `sorting` will know the `quick_sort_algorithm`. Thus the conditional probability table of the node `quick_sort` reflects this kind of dependency by assuming an expert (advanced, beginner, novice) in the parent node `sorting` to be – with high probability – an expert (advanced, beginner, novice respectively) in the child node `quick_sort`. To find such a conditional probability table, we investigated several distributions. Our result can be seen in figure 6.10. In a similar way we found conditional probability tables for weaker dependencies, for example between the child node `gridbag_layout_manager` (the most complicated layout manager in Java 1.1x) and the parent node `layout_manager` (see figure 6.11) and for the root nodes of the BN (see figure 6.12).

layout_manager	P(gridbag=E|.)	P(gridbag=F|.)	P(gridbag=A|.)	P(gridbag=N|.)
E	0.34	0.24	0.16	0.11
F	0.33	0.36	0.24	0.22
A	0.22	0.24	0.36	0.33
N	0.11	0.16	0.24	0.34

Figure 6.11: "weak" conditional probability table for the node `gridbag_layout_manager`, which is dependent on the node `layout_manager`

algorithms	P(algorithms=E)	P(algorithms=F)	P(algorithms=A)	P(algorithms=N)
	0.01	0.01	0.1	0.88

Figure 6.12: Conditional probability table for root nodes

For example, in the CS1 hyperbook, knowledge about control structures is related to knowledge about algorithms (if we observe a user knowing how to implement an algorithm, he will probably have used some operations to control the execution of the program), but nothing can be said if we see a user doing well in building a graphical interface. He might have used some control structures, but not necessarily.

This finishes step two from the construction process of the Bayesian network.

6.3 Advantages of using a BN

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 now a KI_2 with grade expert, advanced, beginner or novice. The Bayesian community states that conditional probabilities are relatively easy to estimate. They go as far as to interpret a conditional probability $P(A|K)$ as "a persons subjective belief in A given a body of knowledge K that may include that person's [particular] assumptions" [93].

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 $\mathcal{KI}$ has already been acquired by a user if the $\mathcal{KI}$ 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. We can use every degree of information about the user's knowledge, not only *failed / not failed*.

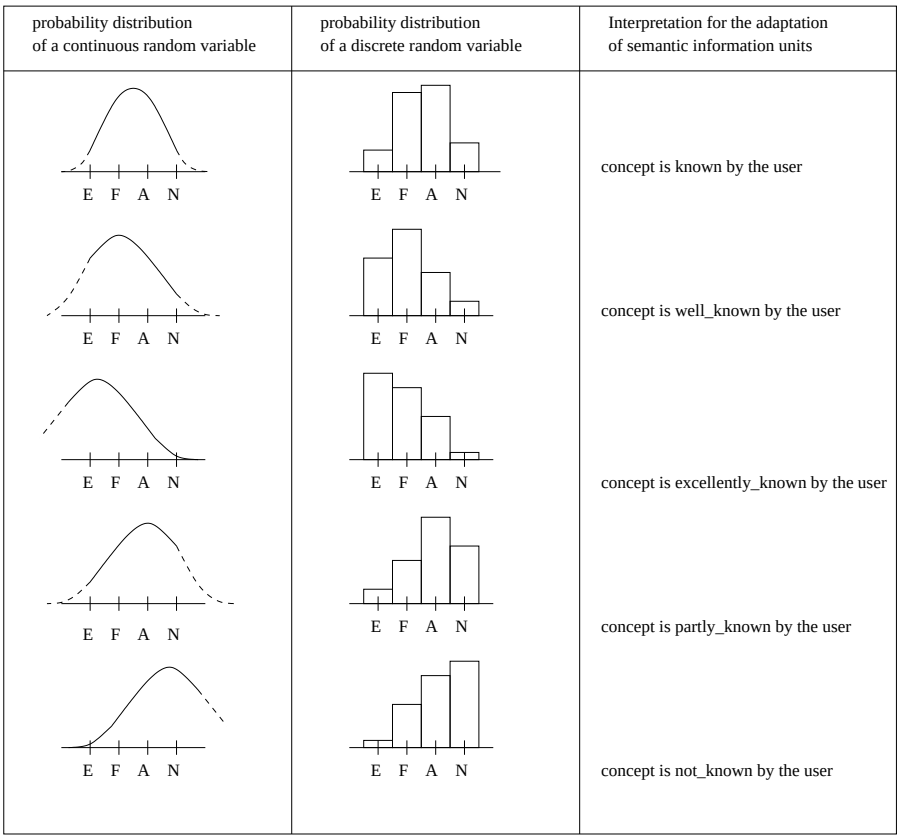

Figure 6.13: Interpretation of the probability distributions given by the Bayesian network

6.4 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 6.13. A knowledge item K is for example "**well_known**" to a user if

$$P(K = \mathbf{E}) + P(K = \mathbf{F}) \geq P(K = \mathbf{A}) + P(K = \mathbf{N}),$$

and it is "excellently_known" by the user, if

$$P(K = \mathbf{E}) \geq P(K = \mathbf{F}) + P(K = \mathbf{A}) + P(K = \mathbf{N}).$$

6.5 The Bayesian Network of the CS1 Hyperbook

So far, we have constructed an acyclic graph on the base of the set of $\mathcal{KI}$s and their learning dependencies (see section 6.1), and have defined suitable conditional probability tables for all kinds of vertices in this graph (see section 6.2).

A part of the Bayesian network of the CS1 hyperbook can be seen in figure 6.14. This Bayesian network contains about 291 nodes. The figure shows the system's estimation about 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.

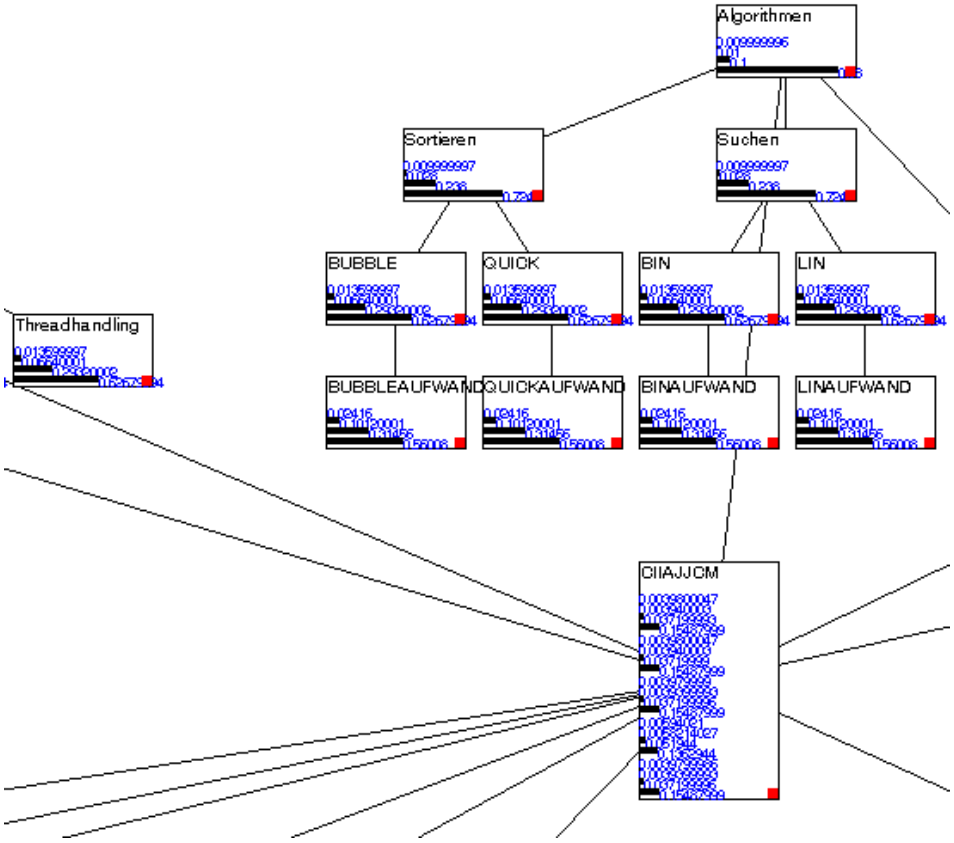

Figure 6.14: Part of the Bayesian network for a CS1 hyperbook

Chapter 7

Enabled Adaptation

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

To solve the following adaptation tasks, the content map (see definition 6) has to be refined. As authors can use arbitrary knowledge items for indexing, we have to find those dependencies in the knowledge model, which are implicitly given in the index. For example, if an author uses a $\mathcal{KI}$ that is not a leaf in the Bayesian network, then this $\mathcal{KI}$ and its children form the index.

Thus, for refining the content map, all index concepts are completed recursively with all of their child concepts. This completion does not affect the levels, only children belonging to the same ACM classification topics (see section 5.1.2) as their parents are added to the index, as can be seen in figure 7.1.

7.1 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

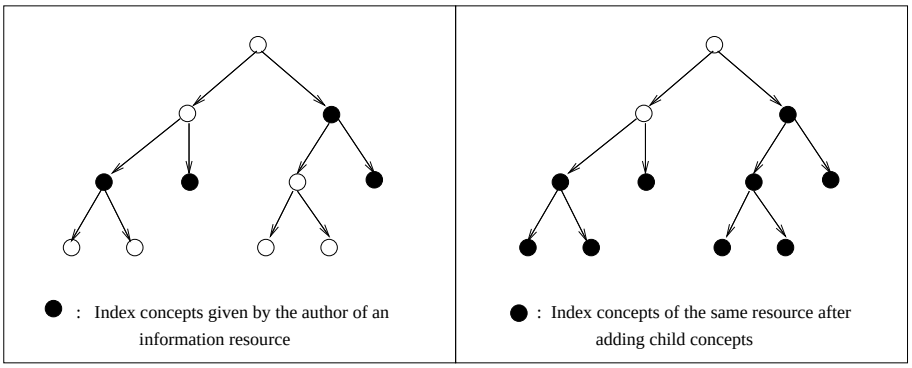

Figure 7.1: Example for the completion of an index set.

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 (see chapter section 4.3). For calculating the relevance of information for a student according to his actual knowledge state, we use a simple *traffic light metaphor* (see section 2.3.3) 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 selecting appropriate information units.

7.1.1 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 $\mathcal{KI}$ is **child_known**, if it is known, well_known, or excellently_known, a $\mathcal{KI}$ is **parent_known**, if it is well_known or excellently_known (see section 6.4).

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* $\mathcal{KI}$s of the index are "child_known":

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

$$\forall K \in I(H) : (\forall C, C \text{ child of } K : C \text{ is child\_known})$$

The ready_for_reading function also checks for those children which are, depending on the three level approach of the Bayesian network, separated from their original parent concepts by the cluster nodes.

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

$$\forall K \in I(H) : (\forall P, P \text{ parent of } K : P \text{ is parent\_known})$$

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.

7.2 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.

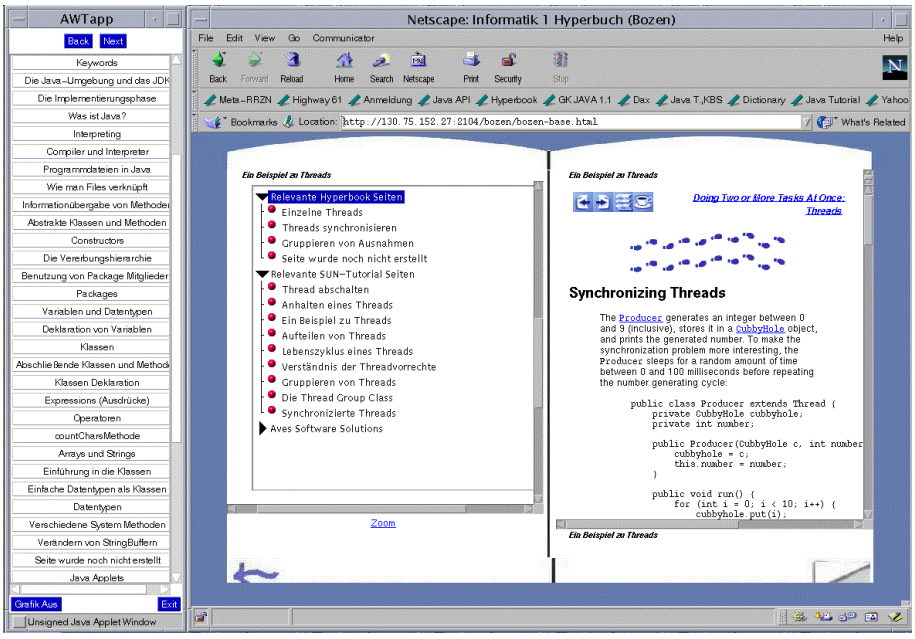

Figure 7.2: The project "synchronizing threads" is presented to a beginner.

7.2.1 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 $\mathcal{KI}$s 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 7.2 shows the project "synchronizing threads" which is presented to a beginner. The trail leading to the required information is therefore very detailed.

7.2.2 Generating a Glossary

Each hyperbook has a glossary which is generated from the knowledge model. A $\mathcal{KI}$ is used as a glossary item, if this $\mathcal{KI}$ is a `what_is*`- $\mathcal{KI}$ (section 5.3.1) or if it is a leaf node in the Bayesian network.

The glossary of the CS1 hyperbook can be seen in figure 7.3

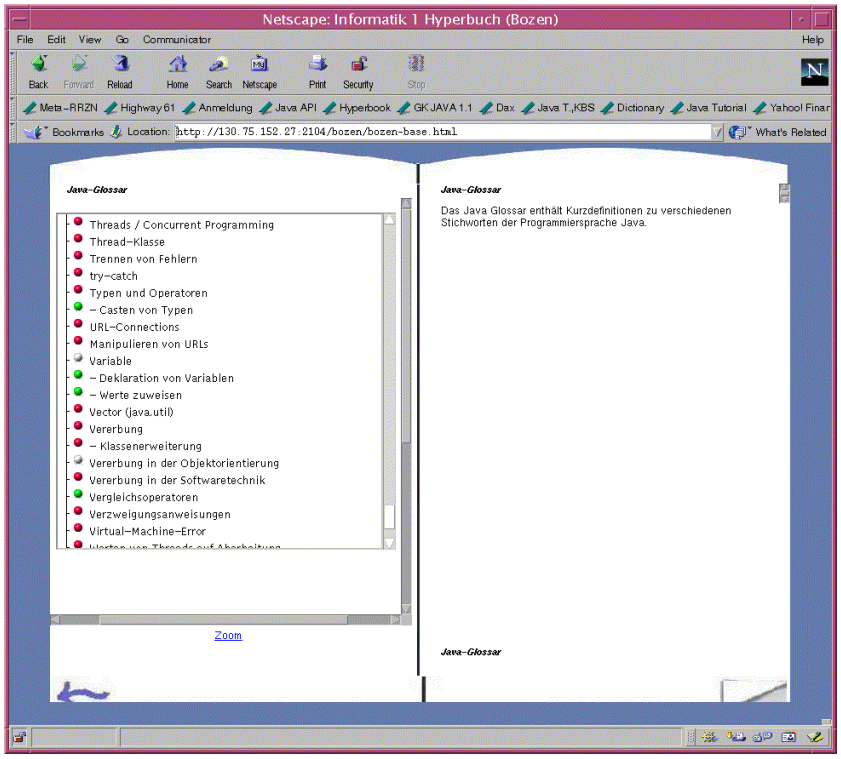

Figure 7.3: The glossary of the CS1 hyperbook

7.2.3 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 base 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 (see section 7.1).

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 pages: 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.

In figure 7.2, we can see the information index belonging to project "synchronizing threads". The index contains the knowledge which is used in the project. Prerequi-

site knowledge required for working on the example is contained in the accompanying trail.

7.3 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 about 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.

7.4 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, as can be seen in figure 7.4. 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.

7.5 Project Based Learning

In order to select suitable projects for a student, the hyperbook contains a project library. Each project is indexed by the $\mathcal{KI}$s 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 [51], 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 $\mathcal{KI}$s comprising the student's goal are sufficiently contained in this project, and

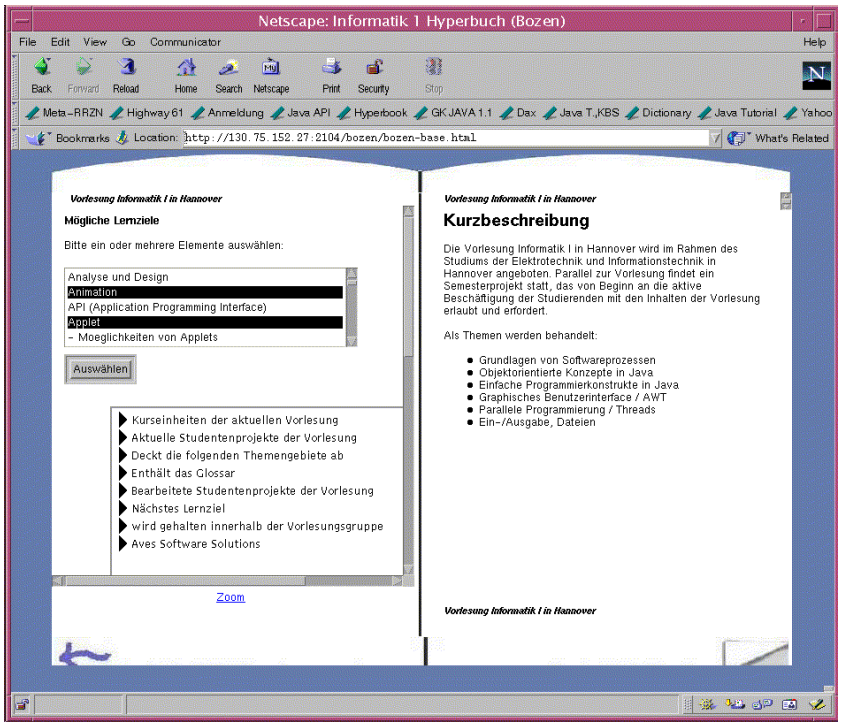

Figure 7.4: A user defines a learning goal

- all $\mathcal{KI}$s which are not part 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 [51]: The first one calculates how good 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*.

7.5.1 Matching: How good fits a project 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 $\mathcal{KI}$s contained in the goal and their relevance in the project.

Each $\mathcal{KI}$ contained in the goal is assumed to have a relevance of 100. The relevance of a $\mathcal{KI}$ for a project is defined by its percentage in relation to the whole project. A short distance means that this $\mathcal{KI}$ is very important for performing the project,

while a large value represents the fact that the $\mathcal{KI}$ is not very relevant for the project. For every $\mathcal{KI}$ 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

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

where relevance(KI, P) is a percental distribution on the index of the project which has to be defined by the project's author (see section 5.3.4).

The project-goal-distance for a project and a given goal is then calculated as the mean value of all these distances:

$$\text{project-goal-distance}(G, P) = \frac{\sum_{KI \in G} \text{distance}(KI, P)}{|G|} \quad . \tag{7.1}$$

7.5.2 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 $\mathcal{U}$ 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.

$$\text{fitness}(G, P, \mathcal{U}) = \frac{\sum_{KI \in I(P) \backslash G} \text{knowledge}(KI, \mathcal{U})}{|I(P) \backslash G|} \quad . \tag{7.2}$$

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

$$\text{knowledge}(KI, \mathcal{U}) = \left(P\,(KI = \mathbf{E}) \cdot 1 + P\,(KI = \mathbf{F}) \cdot \frac{2}{3} + P\,(KI = \mathbf{A}) \cdot \frac{1}{3} \right) \cdot 100.$$

For example, assume that a student has defined a learning goal containing Java applets and animation. The evaluation of the matching algorithm (column "D"), the fitness algorithms (column "F"), and the resulting ordered list (column "No.") of useful projects can be seen in figure 7.5.

The presentation of the selected projects for the learning goal "Java applet and animation" is shown in figure 7.6.

7.6 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 in this portfolio.

No.	D	F	Projektname
0	50	100	EXJTRealityBreakTheSpotApplet
1	50	100	EXJTRunAnimatorAppletTimer
2	50	100	BeispielAnimation
3	73	12	EXJTRunImageSequenceTimer
4	76	12	EXJTRunShapesDemo
5	76	12	EXJTRunSelectionDemo
6	80	10	EXJTASimpleNetworkClientApplet
7	80	12	EXJTRunRectangleDemo
8	80	12	EXJTRunTextXY
9	85	6	EXJTTestDrivinganApplet
10	90	6	EXJTGivingInformationaboutParameters
11	90	10	EXJTThreadsinAppletsTheFullExample
12	90	16	EXJTRunMovingImageTimer
13	95	7	XJTThreadsinAppletsExamples
14	97	11	BeispielCalculator
15	100	4	BeispielVector
16	100	4	EXJTImplementingtheSleeperInterface
17	100	5	BeispielQuickSort
18	100	5	BeispielBugFix
19	100	5	BeispielAuto
20	100	6	EXJTWritingtheCodetoSupportParameters
21	100	7	BeispielAppletGUIAufbau
22	100	10	BeispielNetworking
23	100	10	BeispielCreateUI
24	100	11	BeispielThread
25	100	12	EXJTRunTabDemo
26	100	12	EXJTRunFontDemo
27	100	12	EXJTRunAbsolutePositioningDemo
28	100	12	EXJTRunBoxLayoutDemo
29	100	12	EXJTRunImageDisplayer
30	100	12	EXJTRunCustomLayoutManagerDemo

Figure 7.5: Evaluation of the matching and fitness algorithms for generating a sorted list of projects for reaching the learning goal “Java applets and animation”.

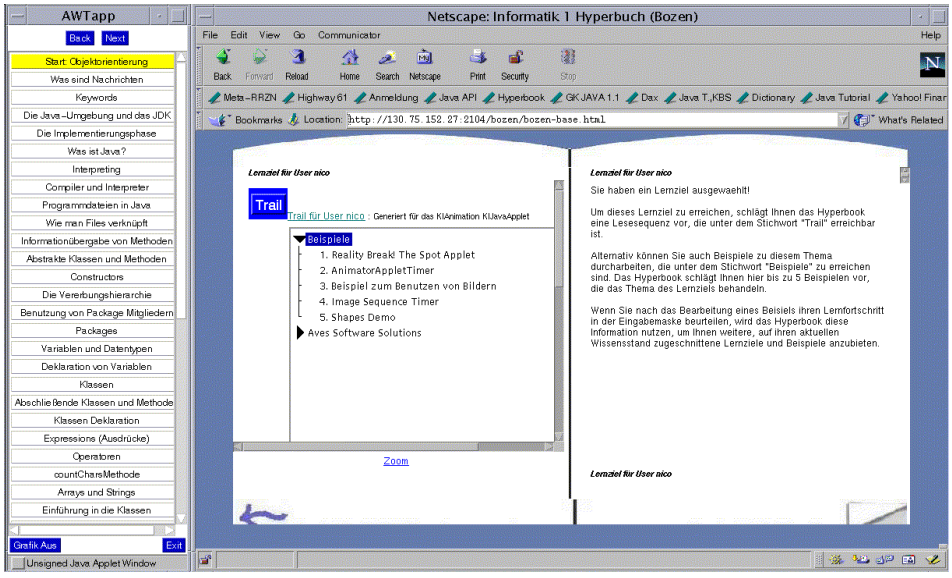

Figure 7.6: Presenting useful projects for reaching the learning goal “Java applets and animation” with an accompanying trail.

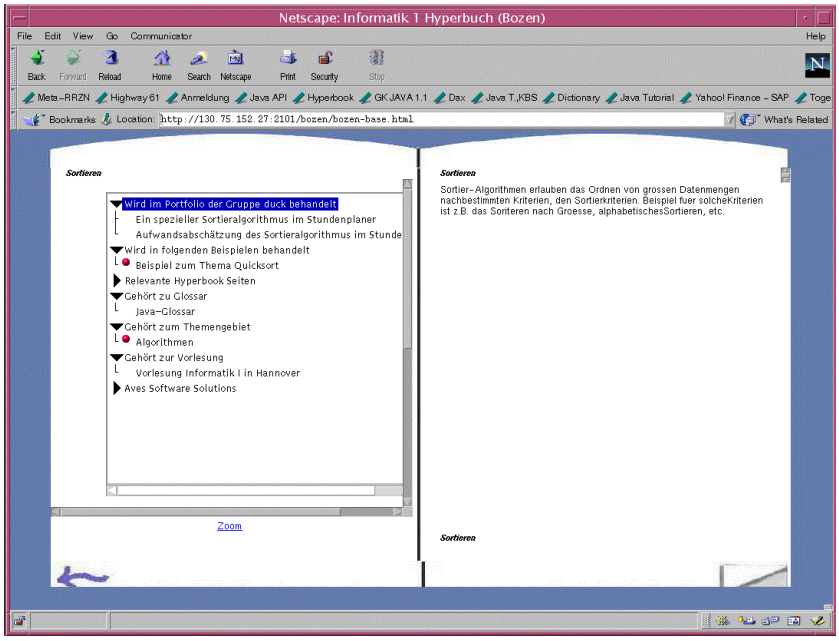

Figure 7.7: Reference to a portfolio

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. This can be seen in figure 7.7, where the glossary item `sorting` has a relation to the corresponding part in the portfolio of “Duck”.

7.7 Project-Based Updating

Figure 7.8 shows a student’s self judgement after he has worked on a project. The project is concerned with foundations of the object oriented programming paradigm. This student judges his knowledge on e.g. `classes` or `objects` expertly but does not feel familiar with `inheritance` yet.

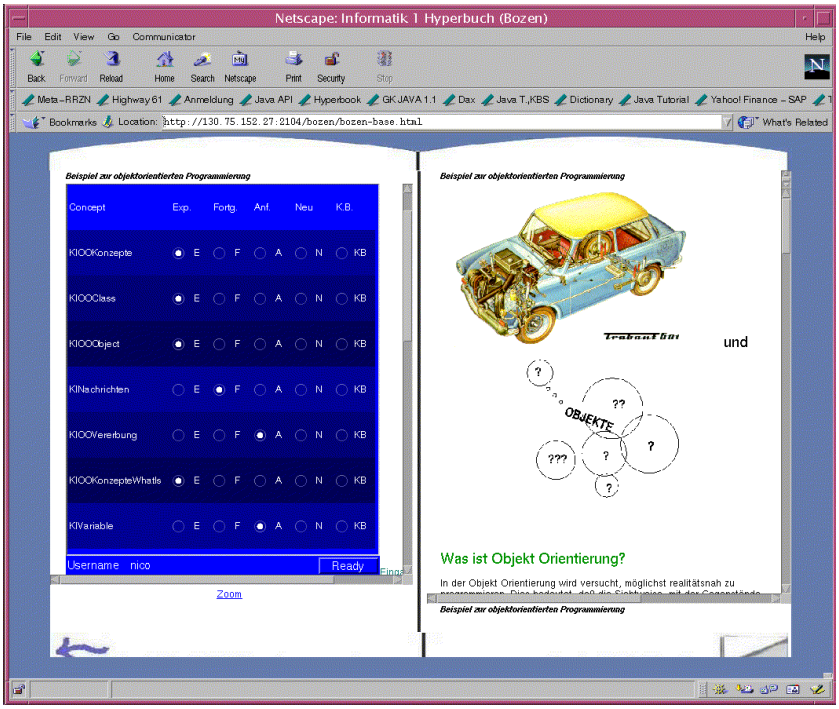

Figure 7.8: Self judgement of a student

Chapter 8

Conclusion

In this work, I have described the concept and realization of the adaptation component for the KBS hyperbook system.

Adaptive hypermedia systems can advantageously be used in education, especially in distance learning. Characteristics of existing adaptive hypermedia systems have been identified to compare and classify the KBS hyperbook system and its adaptation component accordingly.

Based on constructivist learning approaches, I have identified the requirements for building educational applications which benefit from improved teaching strategies. I have shown the advantages of project-based learning and have derived requirements for implementing project-based learning in web-applications.

The general functionality of the KBS hyperbook system with its focus on project-based learning in the internet has been proposed. The use of conceptual models for structuring lectures, courses and information materials has been demonstrated. In addition, the conceptual model has been used for modeling projects and student's portfolios. It is thus an information repository which can be extended by both teachers and learners. The requirements of the adaptation component are also modeled in the conceptual model. I have shown how to index various information units and derived the adaptation functionality in the conceptual model. The general indexing approach underlying the adaptation component enables the KBS hyperbook system to integrate and adapt even distributed information resources. This has been exemplary shown by the integration of the Sun Java tutorial into a hyperbook about programming in Java.

The main part of this thesis is the description of the concept of the adaptation component and its implementation on top of a Bayesian inference mechanism.

In chapter 5, I have described the indexing approach in detail. I have shown, how to structure the index concepts in order to construct a domain knowledge model. The user model, which consists of estimations of the user's knowledge in respect to each concepts in the knowledge model, has been proposed. The resulting adaptation component is robust against changes in the hyperbook: pages can be updated, course information can be changed, etc. In addition, arbitrary information material can be integrated and adapted, and multiple authors can work independently on a hyperbook.

For making inferences about the user's knowledge, I have reimplemented the domain knowledge model from chapter 5 as a Bayesian network. The construction

steps for obtaining such a Bayesian network containing the whole domain knowledge have been described in detail. In order to improve performance of propagating the network, a specialized clustering algorithm has been developed (chapter 6). This YACF-clustering algorithm is tailored to the requirements of our application and enables us to use a Bayesian network which contains the entire domain knowledge. This allows the adaptation component to quickly update its estimation about the user's current knowledge state on base of only few observations. Tricky in the generation of the Bayesian network is, that from observations about the knowledge of advanced topics the knowledge of all prerequisites can be derived. But the lack of knowledge of easy topics decreases the knowledge of advanced topics only moderate.

The way the proposed adaptation component serves hyperbooks to personalize their content and navigational structure has been shown in chapter 7. Hyperbooks can guide their users by generating reading sequences for them, can show next reasonable learning steps to take and propose learning goals. They let their users define learning goals on themselves, give hints to valuable examples, select projects which serve for reaching learning goals or answer information requests, and can select useful information located *anywhere* in the WWW.

The adaptation component and the KBS hyperbook systems have been presented on various workshops and conferences [33, 31, 32, 34, 47, 48, 50, 30, 49, 51, 54, 52, 53, 46, 45] by the author together with her colleagues from the Institut für Rechnergestützte Wissensverarbeitung at the University of Hannover.

The examples of hyperbooks presented in this work can be tested via the URL http://www.kbs.uni-hannover.de/hyperbook/ .

Chapter 9

Outlook

Further work may be concerned with enlarging the indexing functionality of the KBS hyperbook system. At current state, authors have to index their pages themselves. This tasks needs to be supported, semi-automatically, or, at best, automatically.

Therefore I propose to employ information retrieval methods for determining the index of an information resource.

In addition, adaptive hyperbooks should be made more flexible for serving as information repositories even if users have finished their learning. Therefore, the description of the domain knowledge – and thus the index – will be a critical part. The domain knowledge has to be enlarged according to new developments but should, on the other hand, reflect observations about the user's formerly gained knowledge. The idea of exchangeable user models on base of user controlled interfaces can be a step in this direction.

Other interesting extensions would be the modeling of an author's work with a hyperbook, e.g. by supporting the construction of courses and the selection of course materials and examples, including the proposal of material found in the internet, and the visualization of the adaptation component, its estimations and conclusions.

Appendix A

Complete List of the Knowledge Items for the CS1 Hyperbook

```
classes_and_objects
  what_are_classes_and_objects
  class_declaration
    class_modifier
  class_body
    variable
    constants
    variable_declaration
  packages
    what_are_packages
    package_declaration
    import_declaration
  class_instantiation
    life_time_of_objects
    constructors

methods
  what_are_methods
  method_modifier
  method_parameter
    actual_parameter
    fill_ins
  returnvalue
    function
    procedure
  method_declaration

inheritance

interfaces
  what_are_interfaces
  interface_implementation
```

```
  interface_declaration
  multiple_inheritance
  extends_keyword

data_types_and_operators
  what_are_types_and_operators
  primitive_types
  what_are_primitive_types
    class_boolean
      boolean
    class_integer
      int
    class_short
      short
    class_long
      long
    class_character
      char
    class_float
      float
    class_double
      double
    class_string
    class_stringbuffer
    class_system
  casting
  reference_types
    what_are_reference_types
    object
    array
      write_data_into_an_array
      array_declaration
      read_data_out_of_an_array
      multidimensional_arrays
  operators
    what_are_operators
    boolean_operators
    numerical_operators
    coding_of_numbers
    operators_for_comparison
  the_numerical_operators
  statements
    assigning_values
    expression
    identifier

control_structures
  what_are_control_structures
  branching
    if
    switch
  looping
    for
    while
```

```
    do_while
  return
  exit
  block

concurrent_programming_structures
  what_is_concurrent_programming
  single_thread
    what_is_a_single_thread
    start_and_run
    stop
    sleep
  multiple_threads
    what_are_multiple_threads
    synchronization
      what_is_synchronization
      deadlock
      monitor
      wait
      notifyall
    priority

error_handling_and_recovery
  what_is_security
  error_handling
    what_is_error_handling
    security_manager
  security_of_applets
    what_are_security_aspects_for_applets
    possibilities_of_applets
    restrictions
  general_exceptions
    what_are_general_exceptions
    errors
      what_are_errors
      virtual_machine_error
    exceptions
      what_are_exceptions
      try_catch
      exception_handling
        what_is_exception_handling
        giving_exceptions_to_the_caller
        separating_exceptions
        grouping_of_exceptions
      java_exception_hierarchy
      io_exception
    java_virtual_machine
      what_is_the_java_virtual_machine
      instruction_set_of_the_jvm
      object_oriented_concepts_of_the_jvm
      startup_of_the_jvm
      threadhandling_in_the_jvm

java_windowing_system
```

```
  what_are_the_awt_and_gui
  awt_events
  what_are_awt_events
  events
    what_are_events
    action_event
    text_event
    item_event
    adjustment_event
    components
      container
      window
      focus
      key_event
      paint
    mouse_event
  event_model
    what_is_the_event_model
    eventsource
    listener
    adapter
    standard_events
    user_defined_events
    event_model_jdk_1_02
    event_model_jdk_1_1
  graphical_user_interface
    what_is_a_graphical_user_interface
    elements
    layout_manager
      what_is_a_layout_manager
      flow_layout
      border_layout
      grid_layout
      gridbag_layout
      card_layout
    guicontainer
      panel
      frame
      filedialog
    graphics
      what_are_graphics
      manipulating_graphics
        what_is_manipulation_of_graphics
        flashing
        animation
        manipulation

java_networking_constructs
  what_is_networking
  url_connection
    manipulating_urls
  sockets
    what_are_sockets
    serversocket
```

```
    datagramsocket
    multicastsocket
  datagram

algorithms
  what_are_algorithms
  sorting
    what_is_sorting
    costs_of_bubble_sort
      bubble_sort_algorithm
    costs_of_quicksort
      quicksort_algorithm
    searching
      what_is_searching
      costs_of_binary_search
        binary_search
      costs_of_linear_search
        linear_search
      recursion

input_and_output
  what_is_input_output_and_file_handling
  sequential_access_file
  random_access_file
  serialization
  streams
  print

java_applets
  what_are_applets
  appletmethods
  life_cycle
  html_tag
  appletviewer

java_applications
  system_out_println
  input_parameter
  public_static_void_main_method
  executable

java_api
  java_util_package
    what_is_the_java_util_package
    enumeration
    vector
    stack
    hashtable
  java_lang_package
    what_is_the_java_lang_package
    math
    exception
    thread
  java_api
```

```
programming_environment
  what_is_the_programming_environment
  editor
  file
    text_file
    binary_file
    executable_file
    jar_archive
  java_interpreter
    bytecode
    interpreter
  java_compiler
    compiler
  program
  programming
    what_is_programming
    programming_in_java
  garbage_collector
  keywords

object_oriented_design_methods
  object_orientation
    what_is_object_orientation
    oo_class
    oo_object
    oo_message
    oo_inheritance
  software_engineering
    what_is_se
    se_process
      what_is_the_se_process
      se_phases
      simplified_se_process
        what_is_the_simplified_se_process
        se_specification
          what_is_a_specification
          se_subject_method
          se_analyzing_method
          se_generating_a_specification
            requirements_specifiaction
        se_analysis_and_design
          what_is_analysis_and_design
          se_static_analysis_and_design
            what_is_static_analysis_and_design
            se_object_oriented_diagram
              se_classes
              se_attributes
              se_inheritance
              se_relations
          se_dynamical_analysis_and_design
            what_is_dynamical_analysis_and_design
            se_interaction_diagram
              se_scenarios
```

```
      se_interactions
implementation
testing
```

Thus we have 289 knowledge items describing the application domain of the hyperbook. Two cluster nodes (see section 6) are used for generating the graph for the Bayesian Network, which therefore consists of 291 vertices.

List of Figures

Bibliography

[1] Albrecht, D., Zukerman, I., Nicholson, A., and Bud, A. Towards a Bayesian model for keyhole plan recognition in large domains. In *Proceedings of the Sixth International Conference on User Modeling, UM97* (Sardinia, Italy, 1997).

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

[3] Ben-Ari, M. Constructivism in computer science education. In *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Sciene Education* (Atlanta, Georgia, February 1998), ACM Press.

[4] Berners-Lee, T. World Wide Web: An illustrated seminar. Held as an Online Seminar in 1991. http://www.w3.org/pub/WWW/Talks/General.html.

[5] Bibliomania: The Network Library, 1997. http://www.bibliomania.com/.

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

[7] Brusilovsky, P., and Pesin, L. ISIS-Tutor: An intelligent learning environment for CDS/ISIS users. In *Proc. of the interdisciplinary workshop on complex learning in computer environments CLCE'94* (Joensuu, Finland, 1994).

[8] Brusilovsky, P., and Schwarz, E. 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, 1997).

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

[10] Brusilovsky, P., Schwarz, E., and Weber, G. 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).

[11] Bush, V. As we may think. *The Atlantic Monthly 176* (1945), 101–108.

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

[13] Campione, M., and Wallrath, K. *The Java Tutorial*, 2nd ed. Addison Wesley, 1999. http://www.javasoft.com/docs/books/tutorial/index.html.

[14] Cobern, W. W. Contextual constructivism: The impact of culture on the learning and teaching of science. In *The Practice of Constructivism in Science Education*, K. Tobin, Ed. Lawrence Erlbaum Associates, 1993.

[15] Conati, C., Gertner, A. S., VanLehn, K., and Druzdzel, M. J. Online student modeling for coached problem solving using bayesian networks. In *Proceedings of the Sixth International Conference on User Modeling, UM97* (Sardinia, Italy, 1997).

[16] Cooper, G. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence 42* (1990), 393–405.

[17] Dagum, P., Galper, A., and Horvitz, E. Dynamic network models for forecasting. In *Eighth Conference on Uncertainty in Artificial Intelligence* (San Mateo, 1992), Morgan Kaufmann Publishers, Inc., pp. 41–48.

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

[19] de Bra, P. Hypermedia structures and systems: Online Course at Eindhoven University of Technology, 1997. http://wwwis.win.tue.nl/2L690/.

[20] Dechter, R. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. *Artificial Intelligence 41* (1989), 273–312.

[21] Dechter, R., and Pearl, J. Tree clustering for constraint networks. *Artificial Intelligence 38* (1989), 353–366.

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

[23] Duffy, T., and Jonassen, D., Eds. *Constructivism and the Technology of Instruction*. Lawrence Erlbaum Associates, 1992.

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

[25] Eklund, J. Knowledge-based navigation support in hypermedia courseware using WEST. *Australian Educational Computing 11*, 2 (1996).

[26] Engelbart, D. The augmented knowledge workshop. In *A History of Personal Workstations*, A. Goldberg, Ed. Addison Wesley, 1988.

[27] Faulmann, C. *Illustrierte Geschichte der Schrift*. Augustus Verlag, 1980. Originally published in 1880.

[28] Fellbaum, C. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[29] Fink, J., Kobsa, A., and Schreck, J. Personalized hypermedia information provision through adaptive and adaptable systems features: User modeling, privacy and security issues. In *Intelligence in Services and Networks: Technology for Cooperative Competition*, A. Mullery, M. Besson, M. Campolargo, R. Gobbi, and R. Reed, Eds. Springer-Verlag, 1997, pp. 459–467.

[30] Franosch, H., Henze, N., Nejdl, W., and Wolpers, M. How to use a hyperbook model for developing hypertext books. In *Second International Workshop on Knowledge Representation for Interactive Multimedia Systems: Research and Experience KRIMS II* (Trient, Italy, 1998).

[31] Fröhlich, P., Henze, N., and Nejdl, W. Conceptual modeling for educational hyperbooks. In *The 4th International Conference on Multimedia Modelling* (Singapore, Nov. 1997).

[32] Fröhlich, P., Henze, N., and Nejdl, W. Meta-modeling for hypermedia design. In *Proc. of Second IEEE Metadata Conference* (Maryland, Sept. 1997).

[33] Fröhlich, P., Henze, N., and Nejdl, W. Virtual institutes and virtual classrooms: How to enhance teaching and mentoring productivity. In *European Conference on Virtual Enterprises and Networked Solutions* (Paderborn, Apr. 1997).

[34] Fröhlich, P., Henze, N., and Nejdl, W. Hyperbook data modeling. In *International Conference On Electronic Publishing, Document Manipulation and Typography* (Saint Malo, France, Apr. 1998).

[35] Fung, R., and Chang, K. Weighting and integrating evidence for stochastic simulation in Bayesian networks. In *Fifth Conference on Uncertainty in Artificial Intelligence* (Windsor, USA, 1988), Morgan Kaufmann.

[36] Gamper, J., Nejdl, W., and Wolpers, M. Combining ontologies and terminologies in information systems. In *Proc. 5th International Congress on Terminology and Knowledge Engineering* (Innsbruck, Austria, 1999).

[37] Gloor, P. *Elements of Hypermedia Design*. Birkhäuser, 1997.

[38] Goldstein, I. The genetic graph: A represenation for the evolution of procedural knowledge. In *Intelligent Tutoring Systems*, D. Sleeman and J.S.Brown, Eds. Academic Press, 1982.

[39] Good, R. G., Wandersee, J. H., and Julien, J. S. Cautionary notes on the appeal of the new "ism" (constructivism) in science education. In *The Practice of Constructivism in Science Education*, K. Tobin, Ed. Lawrence Erlbaum Associates, 1993.

[40] Grønbæk, K., and Trigg, R. H. *From Web to Workplace: Designing Open Hypermedia Systems*. The MIT Press, 1999.

[41] Guarino, N. Formal ontology and information systems. In *Formal Ontology in Information Systems*, N. Guarino, Ed. IOS Press, 1998.

[42] Guarino, N., Masolo, C., and Vetere, G. Ontoseek: Using large linguistic ontologies for gathering information resources from the web. Tech. Rep. LADSEB-CNR 01/98, National Research Council, LADSEB-CNR, Padova, 1998.

[43] Henrion, M. Propagation of uncertainty in Bayesian networks by probabilistic logic sampling. In *Uncertainty in Artificial Intelligence*, vol. 2. Elvesier, 1988, pp. 149–163.

[44] Henze, N. Auswertung der Befragung der Teilnehmer der Vorlesung Grundzüge der Informatik. Tech. rep., University of Hannover, Feb. 1997.

[45] Henze, N. Evaluation zur Vorlesung Grundzüge der Informatik WS 98 / 99. Tech. rep., University of Hannover, Oct. 1999.

[46] Henze, N., Naceur, K., Nejdl, W., and Wolpers, M. Adaptive hyperbooks for constructivist teaching. *KI-Themenheft 4* (1999).

[47] Henze, N., and Nejdl, W. Eine internet-basierte virtuelle Lernumgebung. *Hannover UNI INTERN 24*, 1 (1997).

[48] Henze, N., and Nejdl, W. 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, Nov. 1997).

[49] Henze, N., and Nejdl, W. Constructivism in computer science education: Evaluating a teleteaching environment for project oriented learning. In *Workshop on Interactive Computer Aided Learning - Concepts and Applications* (Villach, Österreich, Oct. 1998).

[50] Henze, N., and Nejdl, W. Das KBS Virtual Classroom Projekt: Informatik-Ausbildung über das Internet. In *Informatik und Ausbildung* (1998), V. Claus, Ed., Springer Verlag.

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

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

[53] Henze, N., and Nejdl, W. Student modeling in an active learning environment using Bayesian networks. In *Proceedings of the Seventh International Conference on User Modeling, UM99* (Banff, Canada, 1999).

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

[55] Hewitt, J., and Scardamalia, M. 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, April 1996).

[56] Hodges, M. E., and Sasnett, R. M., Eds. *Multimedia Computing: Case Studies from MIT Project ATHENA*. Addison-Wesley, 1993.

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

[58] Höök, K., Karlgren, J., Waern, A., Dahlbäck, N., Jansson, C., Karlgren, K., and Lemaire, B. A glass box approach to adaptive hypermedia. *User Modeling and User Adapted Interaction 6*, 2-3 (1996), 157–184.

[59] Horton, W. *Designing and Writing Online Documentation*. John Wiley & Sons, Inc., 1994.

[60] Horvitz, E., Suermondt, H., and Cooper, G. Bounded conditioning: Flexible inference for dexisions under scare resources. In *Fifth Conference on Uncertainty in Artificial Intelligence* (Windsor, USA, 1989).

[61] Jameson, A. Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User Adapted Interaction 5(3/4)* (1996), 193–251.

[62] Jameson, A. What can the rest of us learn from research on adaptive hypermedia - and vice versa? Tech. rep., University of Saarbrücken, 1999.

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

[64] Jones, K. S., and Willet, P., Eds. *Readings in Information Retrieval*. Morgan Kaufmann, 1997.

[65] Kafai, Y., and Resnick, M., Eds. *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World*. Lawrence Erlbaum Associates, 1996.

[66] Kass, R. Student modeling in intelligent tutoring systems - implications for user modeling. In *User Models in Dialog Systems*, A. Kobsa and W. Wahlster, Eds. Springer, 1989.

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

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

[69] Keep, C., and McLaughlin, T. The Electronic Labyrinth. Tech. rep., The Electronic Labyrinth, 1997. http://jefferson.village.virginia.edu/elab/hfl0038.html.

[70] Knight, K., and Luk, S. K. Building a large-scale knowledge base for machine translation. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)* (Seattle, Washington, Aug. 1994), vol. 1, pp. 773–778.

[71] Kobsa, A. User modeling: Recent work, prospects and hazards. In *Adaptive User Interfaces: Principles and Practice*, M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski, Eds. Elvesier, 1993.

[72] Kobsa, A., and Pohl, W. The user modeling shell system BGP-MS. Tech. rep., University of Konstanz, 1995.

[73] Lamon, M., Chan, C., Scardamalia, M., Burtis, P. J., and Brett, C. 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, April 1993).

[74] Landow, G. P. Course assignments using hypertext: The example of INTERMEDIA. *Journal of Research on Computing in Education 21*, 3 (1989), 349–363.

[75] Landow, G. P., and Kahn, P. Where's the hypertext? The Dickens web as a system-independent hypertext. In *Proceedings of the 4th ACM ECHT Conference on Hypertext* (Milano, Italy, Dec. 1992), ACM, pp. 149–160.

[76] Lauritzen, S., and Spielgelhalter, D. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B 50*, 2 (1988), 157–224.

[77] Lowe, D., and Hall, W. *Hypermedia and the Web.* J. Wiley and Sons, 1999.

[78] Mahesh, K., and Nirenburg, S. A situated ontology for practical NLP. In *Proceedings of the IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing* (Montreal, Canada, Aug. 1995).

[79] Martin, J., and VanLehn, J. OLAE: Progress toward a multi-activity, Bayesian student modeler. In *Proceedings of Artificial intelligence in education AIED* (Edinburgh, Scottland, 1993), pp. 441–417.

[80] McGuinness, D. L. Ontological issues for knowledge-enhanced search. In *Proceedings of the First International Conference on Formal Ontology in Information Systems* (Trento, Italy, June 1998), N. Guarino, Ed., IOS Press, pp. 302–316.

[81] Milosavljevic, M., Tulloch, A., and Dale, R. Text generation in a dynamic hypertext environment. In *Proceedings of the 19th Australian Computer Science Conference* (Melbourne, Australia, January 31 - February 2 1996).

[82] Mislevy, R., and Gitomer, D. The role of probability-based inference in an intelligent tutoring system. *User Modeling and User-Adapted Interaction 5* (1996), 253–282.

[83] Müller, P. Writing hypertext books. Tech. rep., FU Berlin, 1995. http://www.inf.fu-berlin.de/tec/Mosaic/HTB/.

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

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

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

[87] Nelson, T. Replacing the printed word: A complete literary system. In *Proc. IFIP Congress* (Netherlands, 1980), S. Lavington, Ed., pp. 1013–1023.

[88] Nielsen, J. *Multimedia, Hypertext und Internet: Grundlagen und Praxis des elektronischen Publizierens.* Vieweg, 1995.

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

[90] Nyce, J., and Kahn, P. A Machine for the Mind: Vannevar Bush's Memex. In *From Memex to Hypertext: Vannevar Bush and the Mind's Machine*, J. Nyce and P. Kahn, Eds. Academic Press, Inc., 1991.

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

[92] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgen Kaufmann Publishers, Inc., 1988.

[93] Pearl, J. Bayesian decision methods. In *Readings in Uncertain Reasoning*, G. Shafer and J. Pearl, Eds. Morgan Kaufmann, 1990.

[94] Piaget, J. *To Understand is to Invent*. Viking, 1973.

[95] Pradhan, M., Provan, G., Middeton, B., and Henrion, M. Knowledge engineering for large belief networks. In *International Conference on Uncertainty in Artificial Intelligence* (San Francisco, USA, 1994).

[96] Rada, R. *Interactive Media*. Springer, 1995.

[97] Renoult, D. The digital collections of the bibliotheque nationale de france: An experiment on internet. In *IEEE International Conference on the Advances in Digital Libraries* (Washington, USA, 1997).

[98] Resnick, M., and Rusk, N. The computer clubhouse: Preparing for life in a digital world. *IBM Systems Journal 35*, 3-4 (1996), 431–440.

[99] Rich, E. User modeling via stereotypes. *Cognitive Science 3* (1978), 329–354.

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

[101] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[102] Scardamalia, M., and Bereiter, C. Technologies for the knowledge-building discourse. *Commun. ACM 36*, 5 (1993), 37–41.

[103] Schäfer, R., and Weyrath, T. Assessing temporally variable user properties with dynamic Bayesian networks. In *Proceedings of the Sixth International Conference on User Modeling, UM97* (Sardinia, Italy, 1997).

[104] Schank, R., and Cleary, C. *Engines for Education*. Lawrence Erlbaum Associates, 1994.

[105] Sebas: Sony electronic book reader, 1997. http://jefferson.village.virginia.edu/elab/hfl0014.html.

[106] Shachter, R., and Peot, M. Simulation approaches to general probabilistic inference on belief networks. In *Fifth Conference on Uncertainty in Artificial Intelligence* (Windsor, USA, 1989).

[107] Shafer, G., and Pearl, J., Eds. *Readings in Uncertain Reasoning*. Morgan Kaufmann Publishers, Inc., 1990.

[108] Specht, M. Empirical evaluation of adaptive annotation in hypermedia. In *ED-Media and ED-Telekom* (Freiburg, Germany, 1998).

[109] Spiegelhalter, D. A unified approach to imprecision and sensitivity of beliefs in expert systems. In *Artificial intelligence and statistics*. Elvesier, 1989, pp. 47–68.

[110] Steinacker, A., Seeberg, C., Reichenbacher, K., Fischer, S., and Steinmetz, R. Dynamically generated tables of contents as guided tours in aadaptive hypermedia systems. In *Proceedings of the ED-Media Conference* (Seattle, USA, 1999).

[111] The on-line books page, 1997. http://www.cs.cmu.edu/books.html.

[112] von Glasersfeld, E. *Radical Constructivism in Mathematics Education*. Kluwer, 1991.

[113] von Glasersfeld, E. A constructivist approach to teaching. In *Constructivism in Education*, L. P. Steffe and J. Gale, Eds. Lawrence Erlbaum Associates, 1995.

[114] von Glasersfeld, E. *Radikaler Konstruktivismus. Ideen, Ergebnisse, Probleme*. Suhrkamp, Frankfurt, 1996. Originally appeared as: Radical Constructivism. A Way of Knowing and Learning, 1995.

[115] W3C Working Group. W3C resource description framework. http://www.w3.org/TR/PR-rdf-schema/, Oct. 1998.

[116] Weber, G. Episodic learner modeling. *Cognitive Science 20* (1996).

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

[118] Weber, G., and Specht, M. 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).

Lebenslauf

Nicola Henze

Geboren:

25. November 1968 in Hannover, Deutschland

Wissenschaftlicher Werdegang:

Mai 1988	Abitur an der St. Ursula Schule Hannover
Okt. 88 - Feb. 95	Studium der Mathematik mit Nebenfach Informatik, Universität Hannover
28. Februar 1995	Diplom in Mathematik
Aug. 95 - Sep. 96	Wissenschaftliche Mitarbeiterin am Lehrstuhl und Seminar für ABWL, Organisation und Wirtschaftsinformatik, Universität Mannheim
Seit Oktober 96	Wissenschaftliche Mitarbeiterin am Institut für Technische Informatik, Abteilung Rechnergestützte Wissensverarbeitung, Universität Hannover
28. April 2000	Promotion