Modeling Constructivist Teaching Functionality and Structure in the KBS Hyperbook System

Nicola Henze, Wolfgang Nejdl, Martin Wolpers

University of Hannover, Institut für Technische Informatik, Abteilung Rechnergestützte Wissensverarbeitung

Abstract: The KBS Hyperbook System is a system which uses explicit conceptual models and meta data to structure and connect external data. When these external data are pages on the WWW, the corresponding conceptual model takes the role of an information index and determines the navigational structure between these pages (corresponding to one or more views on the external data). The conceptual model also serves as a schema for the integration of new pages (similar to the role of a database schema). In this paper we show how such a model can be used to support three main aspects of constructivist learning in a computer supported teaching environment, namely structural support for goal-oriented learning and projects,the integration of student projects into hyperbook based lecture material and the implementation of student annotations.

Keywords: pedagogical theories for CSCL, supporting constructivistic teaching by explicit conceptual models

Introduction

The KBS Hyperbook System aims to model, organize and maintain open hypermedia systems on the World Wide Web. Open in this context means that these hypermedia systems are able to integrate (possibly distributed) information resources based on World Wide Web standards using a hyperbook metaphor, displaying learning units plus their connections to other units. As hyperbooks are web-applications, they are typically used in distance learning scenarios where a learner / user uses the information from the hyperbook on its own. Thus, it is important to think about useful teaching strategies for enabling a learner / user to actively learn and not only to passively read or "consume" 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 problems/projects and their relationship to information units. Learners can reach learning goals or can receive answers to information requests by working on these problems, which introduce, explain and show the use of the learning items.
This paper starts with a definition of adaptive hyperbooks and a short overview of the modeling approach we have choosen for modeling these hyperbooks. The following section introduces constructivist teaching concepts and identifies and discusses the main aspects we implement in our teaching environment. We will then give a description of our KBS Hyperbook System, and describe how we use the hyperbook system for our CS1 course by modeling the structure of projects and annotations and integrating student projects and annotations into the hyperbook based on portfolios.

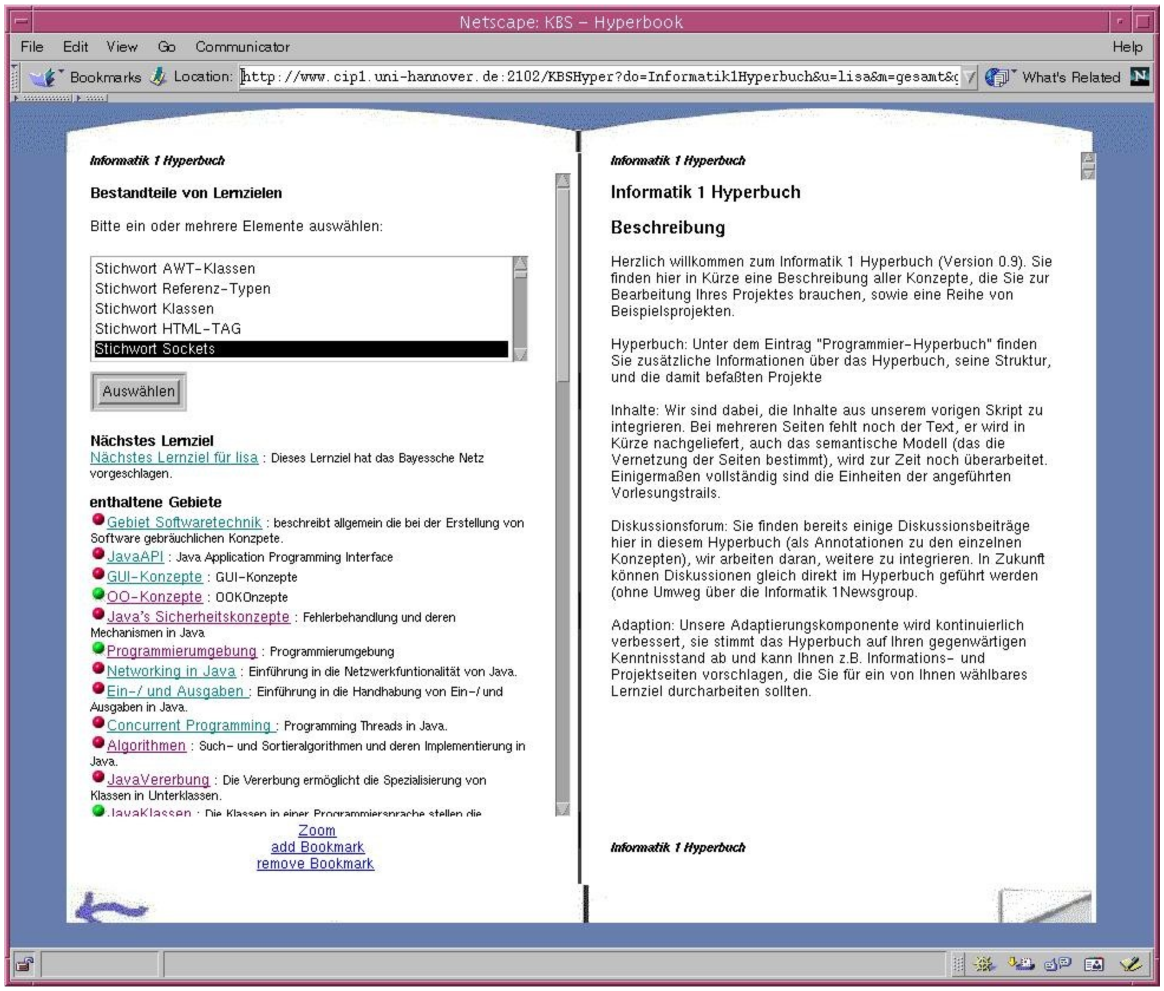
Modeling adaptive hyperbooks

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*, simply characterized as *collections of hypertext documents*. In most cases, these hypertext books still retain the conventional book structure and are partitioned into (sub-) documents called chapters, sections, subsections, or appendices. For the purpose of this article, we focus on a definition of hypertext books which places 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:

> *An adaptive hyperbook for web-based learning is an information repository, which integrates and personalizes a set of (possibly distributed) learning materials (including projects, discussion, etc.) using explicit semantic models and metadata.*

Representation ontology

A very general representation ontology provides the basic constructs from which the structural and domain models are built. This ontology basically defines an extended entity-relationship modeling language (concepts, relations, attributes, inheritance and instantiation) and additional abstractions for index entries referencing the external data objects and visualization concepts (connections, views). This ontology, which is described in more detail in [Nejdl and Wolpers, 1999] is somewhat different from, though compatible with the meta model for RDF (Resource Description Framework, http://www.w3.org/TR/PR-rdf-schema/) schemata. A forthcoming report will describe an import facility from RDF schemata and data into our hyperbook system.



http://www.cip1.uni-hannover.de:2102/KBSHyper?do=Informatik1Hyperbuch&u=lisa&m=gesamt&

Informatik 1 Hyperbuch

Bestandteile von Lernzielen

Bitte ein oder mehrere Elemente auswählen:

Stichwort AWT-Klassen
Stichwort Referenz-Typen
Stichwort Klassen
Stichwort HTML-TAG
Stichwort Sockets

Auswählen

Nächstes Lernziel
Nächstes Lernziel für lisa : Dieses Lernziel hat das Bayessche Netz vorgeschlagen.

enthaltene Gebiete
- Gebiet Softwaretechnik : beschreibt allgemein die bei der Erstellung von Software gebräuchlichen Konzpete.
- JavaAPI : Java Application Programming Interface
- GUI-Konzepte : GUI-Konzepte
- OO-Konzepte : OOKOnzepte
- Java's Sicherheitskonzepte : Fehlerbehandlung und deren Mechanismen in Java
- Programmierumgebung : Programmierumgebung
- Networking in Java : Einführung in die Netzwerkfuntionalität von Java.
- Ein-/ und Ausgaben : Einführung in die Handhabung von Ein-/ und Ausgaben in Java.
- Concurrent Programming : Programming Threads in Java.
- Algorithmen : Such- und Sortieralgorithmen und deren Implementierung in Java.
- JavaVererbung : Die Vererbung ermöglicht die Spezialisierung von Klassen in Unterklassen.
- JavaKlassen : Die Klassen in einer Programmiersprache stellen die

Zoom
add Bookmark
remove Bookmark

Informatik 1 Hyperbuch

Informatik 1 Hyperbuch

Beschreibung

Herzlich willkommen zum Informatik 1 Hyperbuch (Version 0.9). Sie finden hier in Kürze eine Beschreibung aller Konzepte, die Sie zur Bearbeitung Ihres Projektes brauchen, sowie eine Reihe von Beispielsprojekten.

Hyperbuch: Unter dem Eintrag "Programmier-Hyperbuch" finden Sie zusätzliche Informationen über das Hyperbuch, seine Struktur, und die damit befaßten Projekte

Inhalte: Wir sind dabei, die Inhalte aus unserem vorigen Skript zu integrieren. Bei mehreren Seiten fehlt noch der Text, er wird in Kürze nachgeliefert, auch das semantische Modell (das die Vernetzung der Seiten bestimmt), wird zur Zeit noch überarbeitet. Einigermaßen vollständig sind die Einheiten der angeführten Vorlesungstrails.

Diskussionsforum: Sie finden bereits einige Diskussionsbeiträge hier in diesem Hyperbuch (als Annotationen zu den einzelnen Konzepten), wir arbeiten daran, weitere zu integrieren. In Zukunft können Diskussionen gleich direkt im Hyperbuch geführt werden (ohne Umweg über die Informatik 1Newsgroup.

Adaption: Unsere Adaptierungskomponente wird kontinuierlich verbessert, sie stimmt das Hyperbuch auf Ihren gegenwärtigen Kenntnisstand ab und kann Ihnen z.B. Informations- und Projektseiten vorschlagen, die Sie für ein von Ihnen wählbares Lernziel durcharbeiten sollten.

Informatik 1 Hyperbuch

Figure 1: Entry page of CS1 hyperbook with concepts (right frame) and relations (left frame)

Different forms of information units correspond to concepts and their aspects/attributes and can be displayed in a WWW browser. The navigational structure between these concepts is based on the relations between them. Together, concepts and relations define the way in which information is presented to the reader. Our current hyperbook system separates the display area of a regular WWW-browser into two equally sized frames for displaying attributes and their content and relations: the right frame displays the textual and image data represented by concepts and their attributes; the left frame shows the concept's relations in the form of annotated hypertext-links. Fig. 1 shows as an example the main entry page of our CS1-hyperbook.

Supporting Constructivist Learning

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 [von Glasersfeld, 1996]) and its applications for teaching (e.g. [von Glasersfeld, 1995]). In [von

Glasersfeld, 1996], 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 contextually and socially oriented view of constructivism (see for example the discussion in [Cobern, 1993]). A less ambitious definition just acknowledges, that learners (including scientists) must construct and reconstruct their own meaning for ideas about how the world works ([Good et al., 1993]), 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 [Piaget, 1973]) 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 then 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.

Constructivism and Teaching

In [von Glasersfeld, 1995], 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/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 rote-learned "right" answers.
Constructivist concepts discussed in the papers from [von Glasersfeld, 1991] 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. Papert and his colleagues [Papert, 1993,Kafai and Resnick, 1996,Resnick and Rusk, 1996], who use the term *constructionism* to especially stress learning as a (social) design activity, build heavily upon 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 projects. 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 there own projects, an open learning community with mentors, students, students as mentors and open projects. 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 [Lamon et al., 1993]) and its successor *Knowledge Forum* (see e.g. [Scardamalia and Bereiter, 1993,Lamon et al., 1993,Hewitt and Scardamalia, 1996]), 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 knowledge map). It focuses on *intentional learning*, where learners strive to expand their knowledge collectively.

Implementing Constructivist Teaching Concepts in a CS1 Course

Based especially on the ideas of learning as design activity (as advocated by Papert and colleagues) and learning as an intentional activity involving knowledge-building and discussion (as in CSILE), we focus in our CS1 course on the following three issues:

- integrating goal-oriented learning and projects (authored by lecturers and students) into our course materials
- connecting student projects with the rest of the course material to build up portfolios [Duschl and Gitomer, 1991] showing which CS1 concepts have been applied (and thus learned) to which part of the project
- modeling student annotations (such as tips, questions and answers) as part of the course material

The structural model of our hyperbook concentrates on this problem-oriented and inquiry-oriented aspect and explicitly models these relevant aspects to support an goal-directed and inquiry-oriented learning style. Students need to know, which materials are necessary for specific projects, and use (personalized) learning sequences and indices to retrieve the required information and hyperbook pages. Goal orientation is an important aspect of our educational hyperbooks. Since we don't 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 thus their own learning sequence. In each step they can ask the hyperbook for relevant material, teaching sequences and hints of practice examples and projects. If they need advice to find their own learning path, they can ask the hyperbook for the next suitable learning goal.

The "Introduction to Java Programming" hyperbook

In our teaching environment, we are currently using our KBS Hyperbook System for several of our courses, the largest one (with about 250 students in the last semester) being a CS1 course "Introduction to Programming". The course is based on constructivist teaching concepts, and builds heavily on project work as well as discussion of problems and solutions by the students.

Structural modeling for adaptive hyperbooks

Central to the structural model (fig. 2) is the concept of a *semantic information unit* (SIU) whose instantiations contain the main information units contained in the hyperbook. The set of SIUs is used for modeling the application domain. Relationships between SIUs are modeled by several semantic relations, which structure the knowledge referenced by the SIUs, for example to relate general knowledge to specializations, related concepts, etc. Semantic structures that emerge for domain modeling are e.g. taxonomies based on inheritance hierarchies and more general domain ontologies including arbitrary relations.

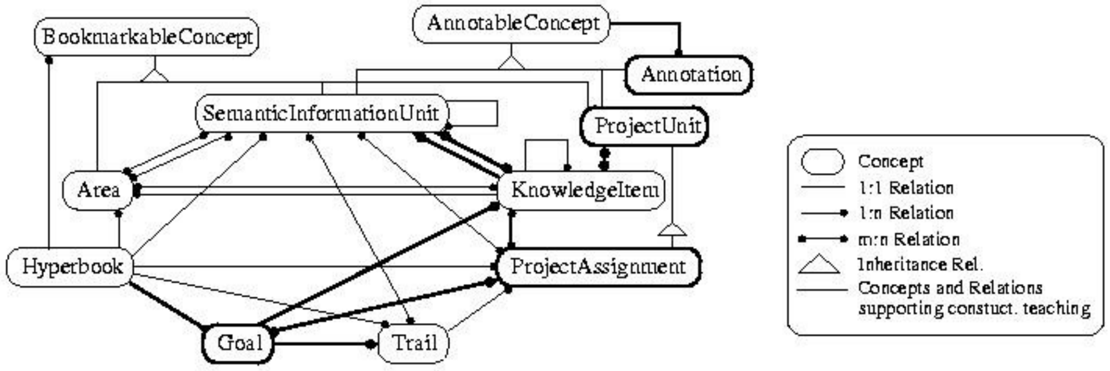


Figure 2: Main concepts of the structural model emphasizing concepts and relations supporting constructivist teaching

To find SIUs in a hyperbook, we specify the main topics of the book. The hyperbook for our course "Introduction to Java Programming" contains main level topics, which we have related to the ACM Computing Classification System (1998 version, http://www.acm.org/class/1998/ccs98.html) [Henze and Nejdl, 1999a]. We use these main topics to group the contents of a hyperbook in areas: each main topic corresponds to an area in the hyperbook (fig. 1). Thus a student can choose several entry points to the hyperbook. Hints for useful entry points - according to the student's actual knowledge state - are given by annotating the links to areas using traffic lights (see section 4).

Modeling learning dependencies

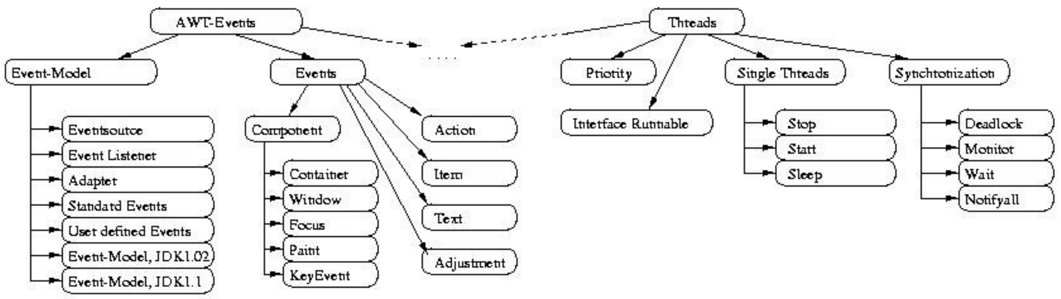


Figure 3: Part of KI dependency hierarchy for the CS1-Hyperbook

To allow several authors to define their individual domain models - this is especially important if we want to enable several teachers to build their own navigational and/or conceptual structure on the teaching material - a second model of the domain is used for indexing of concepts. This second model is based on *knowledge items* (KIs) which denote either elementary knowledge concepts of the application domain, for example the "if"- or "while"-concepts in a programming language, or compound concepts, like "knowledge about flow control statements". All KIs are connected in a dependency hierarchy (a polytree, a part which can be seen in figure 3) and thus form a hierarchical overview about the knowledge contained in the hyperbook. This decoupling of knowledge item model and domain models provides independence of the actual applied domain model and makes the system robust against changes in either the domain model or the content of SIUs which may vary from author to author. The KI model is also used by a Bayesian network for user adaptation [Henze and Nejdl, 1999b].

Integration of examples and projects

In order to support project-oriented teaching as described in section 3, our conceptual model contains the concept *project unit* and subclasses thereof, such as *project* and *project assignment* (PA). Project units contain project descriptions, and different parts of a (student) project. In order to enable students to integrate their projects in the hyperbook in a structured and meaningful way, we model a project as a part-whole hierarchy representing the different parts of each student project, shown in figure 4.

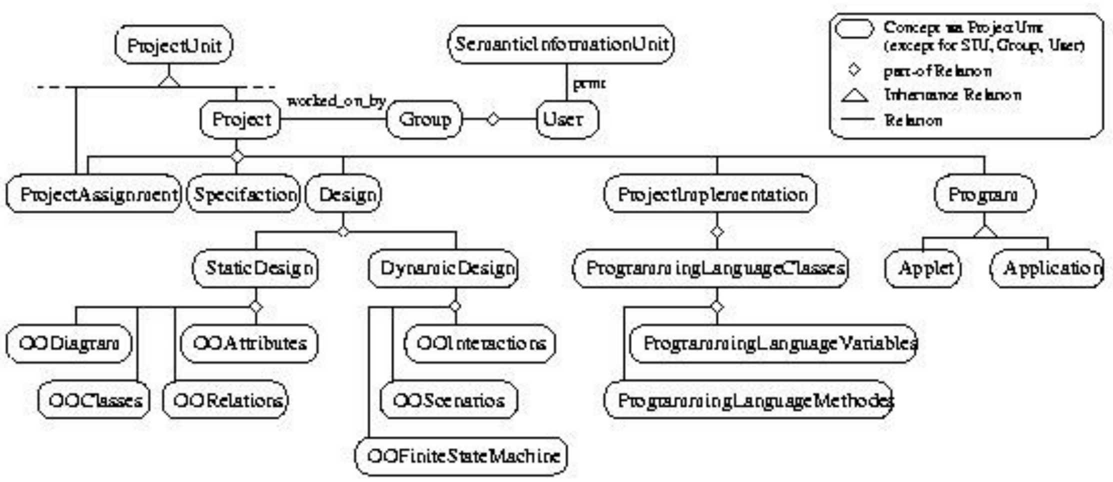


Figure 4: Schematic view of the ProjectUnit part-whole hierarchy

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, the documentation of the implementation and the program code itself. The program code is broken down into different (Java) classes, with each class describing its attributes and methods. See figure 5 as an example how a student group named BugFix modeled their project according to the ProjectUnit hierarchy.

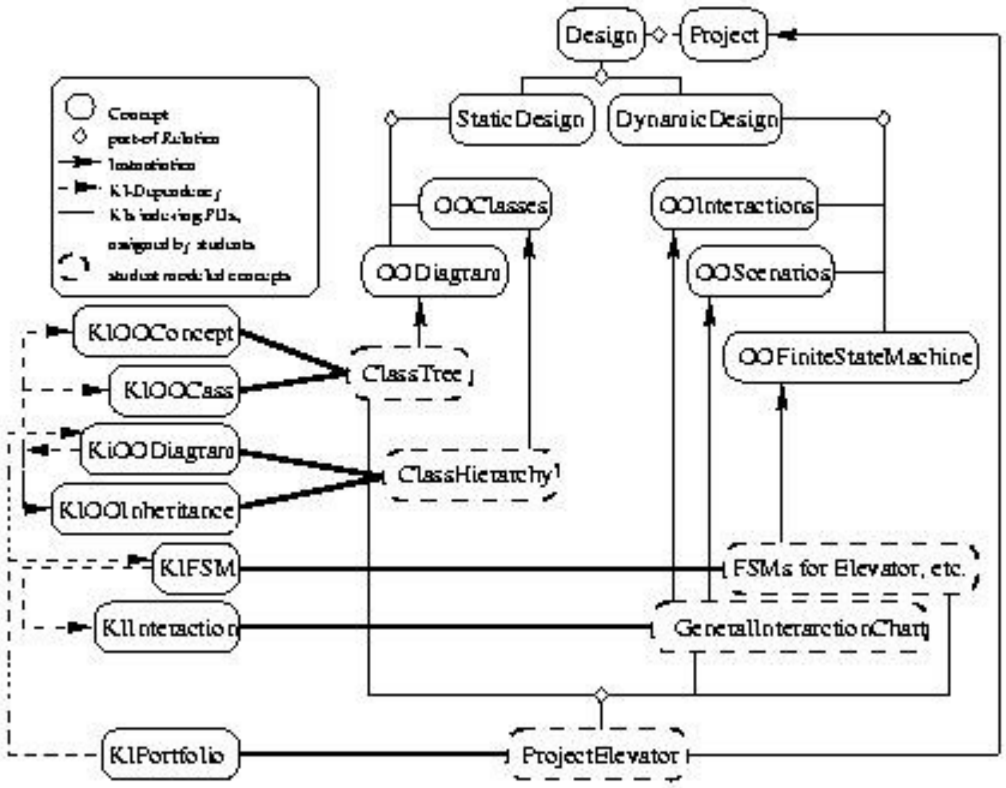


Figure 5: Example of the ProjectUnit hierarchies used by a student group named BugFix with attached KIs

Portfolios: documentation of student projects

As discussed in [Duschl and Gitomer, 1991], assessment based on portfolios is based on the idea, that project results can be used to represent and assess which concepts a student has successfully applied / learned. Thus a portfolio can be modeled by a set of relations between parts of the project and the corresponding concepts which have been used for these subprojects (see figure 5). In our conceptual model, this is expressed by a relationship between project units and knowledge items. In general, we include the root concepts in the polytree of the KIs in this subset. However, a more detailed portfolio is possible by exchanging these high level KIs with lower level ones.
To give another example, inheritance is an important aspect of object oriented programming and therefore is part of our portfolio. If students understand and successfully apply inheritance in their project we can safely assume, that they know how to extend classes, overload methods, etc. Therefore the students connect some of their classes to the KI "inheritance" and need not specifically connect to the KIs "the extends keyword", "overwritten methods", etc. For modeling the concept of software engineering, more details are required to emphasize that these parts should be contained in the project description (e.g. an object oriented diagram, object oriented analysis, etc.). In this case, we do not use the root concept "software engineering" (which is very general indeed), but use instead the lower level KIs "object oriented diagram", "specification", etc. In this way, we define both the basic structure of student projects as well as their connection to the remainder of the course material (see figure 9 for an example). Different project parts are described on different pages, for the implementation part the program javadoc is used which

splits Java code into classes with attributes and methods (as defined by our model, figure 5).

Guided Tours and Learning Goals

To support inquiry-oriented work with the hyperbook, *trail*- and *goal*- concepts are used for providing access to the knowledge contained in the hyperbook. Trails are basically sequences of SIUs. They are either generated and therefore tailored to the student's actual knowledge (see section 4) or are predefined (e.g. for usage during a lecture). The goal concept supports the student in defining her/his own particular learning goals. The student can choose her/his learning goal by defining a set of KIs as a goal. This subset is used by the student modeling component to determine relevant hyperbook-pages for this goal: a list of appropriate project assignments and a trail. This trail consists of a sequence of those SIUs that contain relevant information for reaching the chosen goal.

From a SIU the student has access to appropriate project assignments (PAs), goals (if the student has already defined some) and trails. The student therefore can view the application of a SIU knowledge on a project page, or enter a trail. The main entry point for students is the *hyperbook* concept, which is related to all areas, PAs, etc. (similar to a table of contents). SIUs also participate in a student defined relationship called *bookmarks*, thus implementing bookmarking functionality. Students can continue with one of these concepts, define a learning goal or select a predefined trail.

Enabling discussions and annotations

To give hyperbook users the possibility to discuss the contents of the hyperbook and to encourage feedback to the authors, a simple annotation system has been integrated into the hyperbook. This annotation system allows the user to add annotations and to read existing annotations. The annotations are shown as structured discussions in the hyperbook. In the current version a simple model for structuring of annotations is being used (fig. 6). The user has the possibility to give tips, ask questions or criticize the contents or to answer to certain questions. The following shows the modeling of the annotations.

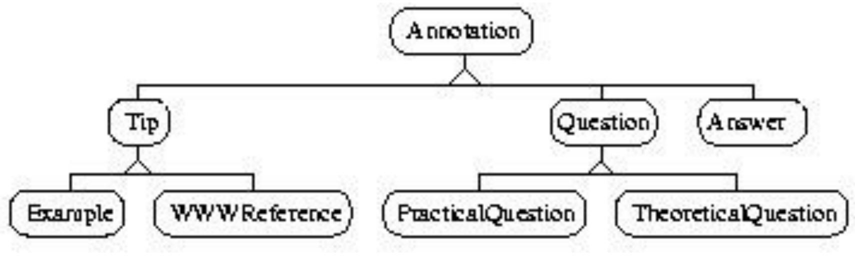


Figure 6: Schematic view of the annotation hierarchy

Annotations are defined as concepts in the hyperbook and therefore are handled and visualized as all other existing concepts. In addition to the attributes of a concept an annotation contains attributes like date and author. All annotatable concepts can be annotated. Annotations are also defined as annotatable concepts and therefore they can be annotated as well. To compose an annotation the user is given an HTML-form. The content of an annotation is stored in in the file system.

Technical realization

The KBS-Hyperbook system is implemented entirely in Java. A servlet residing in the Java Web Server (fig. 7) 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 functionality such as trails is also realized by Java client Applets.

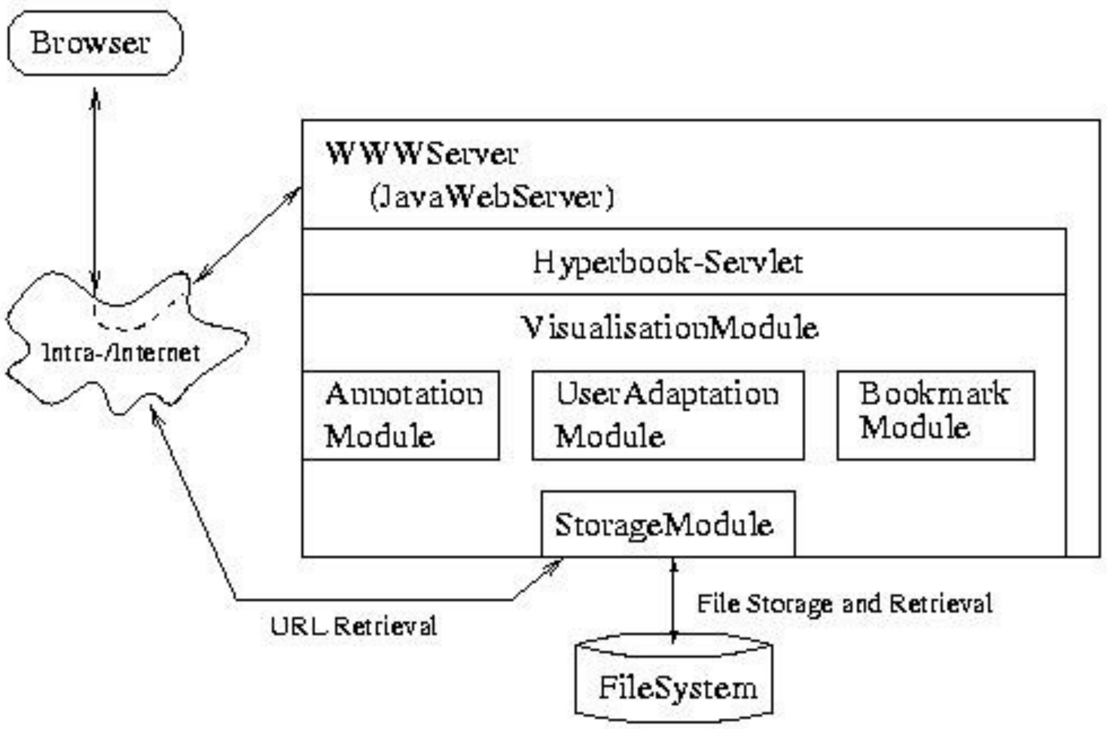


Figure 7: Schematic view of the implementation of the hyperbook system

Conclusion

This paper discussed the idea of supporting different aspects of constructivist teaching in the KBS Hyperbook System by supporting structural models for various aspects of such an approach. The system is being built for a constructivist learning environment and emphasizes project-oriented and goal-driven learning. We have described the use of the system in our course "Introduction to Java Programming" and the integration of teacher and student materials and projects in the form of portfolios.

Future projects will use the system to integrate and connect course material from different courses and different universities by suitable metadata annotations and classifications to interconnect materials and projects in a single repository. Additional work will concentrate on adapting these conceptual models and metadata schemata for the special needs of different teachers and on an improved integration of material from arbitrary WWW sources (based on RDF).

Bibliography

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

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

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

Henze, N. and Nejdl, W. (1999). Bayesian Modeling for Adaptive Hypermedia Systems. In *ABIS99, 7. GI-Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, Magdeburg, September 1999. http://www-mmt.inf.tu-dresden.de/joerding/abis99/EndPaper/henze_16/henze_16.html

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

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

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

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

Nejdl, W. and Wolpers, M. (1999). KBS hyperbook - a data-driven information system on the web. In *Proceedings of the 8th International World Wide Web Conference*, Toronto.

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

Piaget, J. (1973). *To Understand is to Invent*. Viking.

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

Scardamalia, M. and Bereiter, C. (1993). Technologies for the knowledge-building discourse. *Communications of the ACM*, 36(5):37-41.

von Glasersfeld, E., editor (1991). Radical Constructivism in Mathematics Education. Kluwer.

von Glasersfeld, E. (1995). A constructivist approach to teaching. In Steffe, L. P. and Gale, J., editors, *Constructivism in Education*. Lawrence Erlbaum Associates.

von Glasersfeld, E. (1996). *Radikaler Konstruktivismus. Ideen, Ergebnisse, Probleme*. Suhrkamp, Frankfurt. originally appeared as: Radical Constructivism. A Way of Knowing and Learning, 1995.

Authors' addresses

Nicola Henze (henze@kbs.uni-hannover.de)

Wolfgang Nejdl(nejdl@kbs.uni-hannover.de)

Martin Wolpers (wolpers@kbs.uni-hannover.de)

University of Hannover; Institut für Technische Informatik, Abteilung Rechnergestützte Wissensverarbeitung; Appelstr. 4; D-30167 Hannover. Tel. +49 511 762-19711. Fax +49 511 762-19712.