Modeling Constructivist Teaching Functionality and Structure in the KBS Hyperbook System

Nicola Henze and Wolfgang Nejdl and Martin Wolpers
Institut für Rechnergestützte Wissensverarbeitung
University of Hannover
Lange Laube 3, 30159 Hannover, Germany
email: {henze,nejdl,wolpers}@kbs.uni-hannover.de

April 30, 1999

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 two main aspects of constructivist learning in a computer supported teaching environment, namely the integration of student projects into hyperbook based lecture material and the implementation and visualization of student annotations.

Keywords: modeling functionality of the system, supporting constructivist teaching by explicit conceptual models

1 Introduction

Previous approaches like RMM [Isakowitz et al., 1995] have introduced the notion of semantical modeling for hypertext collections, based on database oriented modeling techniques. We generalize these approaches in our KBS Hyperbook System by decoupling meta data / conceptual models, which explicitly model all relevant units of the hyperbook, and data / document units referencing the actual data (comparable to the idea of indexing discussed in [Niederée et al., 1998]), and implementing a meta data-based system using several sets of abstractions to visualize document units and their (semantic) relationships. As a modeling language, we use the language O-Telos [Mylopoulos et al., 1990], which is an object-oriented design language with additional deductive rules and constraints, providing a rich set of meta-modeling facilities.

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 (based on Java)”. 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.

In the first two sections we will give a short overview of constructivist teaching concepts and identify and discuss the main aspects we implement in our teaching environment. We will then give a short 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 these conceptual models.1

1We are using ontologies in the wider sense discussed in [Guarino, 1998], which includes 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 the contents of our hyperbook based on its contents. To avoid misunderstandings, we use the term conceptual models whenever it is appropriate.
2 Supporting Constructivist Learning

2.1 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 context 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.

2.2 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 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, 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.

2.3 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 the 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. As 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 (restricted only by inherent dependencies between parts of the material). 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 learning path they can ask the hyperbook for the next suitable learning goal.

In the following we will discuss the basic architecture of the KBS Hyperbook System (based on explicit modeling of all relevant aspects of the book) and on the specific conceptual models used to support the constructivist learning concepts discussed so far.

Let us now briefly describe the system functionality and the system itself as seen from a users point of view. The KBS Hyperbook System structures and displays hypertext materials based on object-oriented conceptual models / ontologies. As mentioned before, these models are expressed in the modeling language O-Telos [Mylopoulos et al., 1990], an object-oriented design language with additional deductive rules and constraints. The models are used as meta data for structuring and connecting external data like files in the local file system or URLs on the Intra-/Internet.

![Figure 1: The general representation ontology](image-url)
3 The KBS Hyperbook System

A very general representation ontology (comparable to the frame ontology used in Ontolingua, [Gruber, 1993]) 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). Figure 1 shows this ontology, which is described in more detail in [Nejdl and Wolpers, 1999].

![Figure 2: The entry page of the CS1 hyperbook](image)

Concepts and their attributes correspond to different forms of information units which 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 concepts relations in the form of annotated hypertext-links. Figure 2 shows as an example the main entry page of the CS1 hyperbook.

Often simple hypertext links do not provide enough information for the user to decide whether to follow the link or not. Therefore each link is accompanied by a heading, a short abstract of the concept it links to and in many cases an annotation (suggesting further reading or not, based on student knowledge). We use a simple traffic light metaphor for annotation ([Weber and Specht, 1997, Brusilosky and Schwarz, 1997]). A red ball indicates that the user is probably not ready for following
Figure 3: The page of the group BugFix of the elevator project

the link, a grey color indicates already learned concepts, a green color suggests appropriate information. As an example, figure 3 shows how the links are displayed with short abstracts and annotations, grouped by the types of relations.

Figure 4: Schematic view of the implementation of the hyperbook system

The KBS-Hyperbook system is implemented entirely in Java. A servlet residing in the Java Web Server (see figure 4) represents the whole system. The user 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.
4 Conceptual Models for Supporting Constructivist Teaching

4.1 Domain Related Modeling Concepts and Indexing

Structuring concepts in domain ontologies has been an important activity for example in natural language processing ([Fellbaum, 1998, Mahesh and Nirenburg, 1995, Knight and Luk, 1994]) and information retrieval ([Guarino et al., 1998, McGuinness, 1998, Benjamins and Fensel, 1998]), and is related to work in terminological research (for a comparison see e.g. [Gamper et al., 1999]). In this paper we will concentrate on the abstraction level above.

Central for this part of our conceptual model (see figure 5) 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 (see figure 5, relation 2), 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 for example taxonomies based on inheritance hierarchies and more general domain ontologies including arbitrary relations (see e.g. [Gruber, 1994]).

SIUs can be grouped into areas, which do not model knowledge of the application domain but just group SIUs into different sets. An introduction of an area is displayed when a user enters an area concept.

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 6) and thus form a hierarchical overview about the knowledge contained in the hyperbook. This decoupling of knowledge item model and domain models provides

![Figure 5: Overview of the structural model](image-url)
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 change from author to author. The KI model is also
used by a Bayesian network for user adaptation [Henze and Nejdl, 1999].

4.2 Project Related Modeling Concepts

In order to support project-oriented teaching as described in section 2.3, our conceptual model contains
the concept project unit and subclasses thereof, such as project and project assignment(PA). Project units
contain contains project descriptions, and different parts of a (student) project.

Figure 6: Part of KI dependency hierarchy for the CS1-Hyperbook

Figure 7: Schematic view of the ProjectUnit part-whole hierarchy

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,
as shown in figure 7. 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.

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. In our conceptual model, this is
expressed by a relationship between project units and knowledge items. Only a subset of knowledge
items is used in such a portfolio. 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 an 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. 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).

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 ([Henze and Nejdl, 1999]) or are predefined (e.g. for usage during a lecture). The goal concept supports the student in defining his/her own particular learning goals. The student can choose his/her learning goal by defining a set of KIs as a goal (see figure 5, relation 4). This subset is used by the system to determine relevant hyperbook-pages for this goal: a list of appropriate project assignments (PAs, figure 5, relation 15) and a trail (see figure 5, relation 14). This trail consists of a sequence of those SIUs (figure 5, relation 13) that contain relevant information for reaching the chosen goal.

From an SIU the user has access to appropriate PAs, goals (if the user has already defined some) and trails. The user therefore can view an application of the knowledge 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 SIUs, areas, PAs, etc. (similar to a table of contents). It also participates in a student defined relationship called bookmarks, thus implementing bookmarking functionality. Students can continue with one of these concepts, define a learning goal of select a predefined trail. To denote ownership of projects and project pages, group and user concepts model student groups and their members.

4.3 Discussion Related Modeling Concepts

![Annotation Hierarchy](image)

Figure 8: Schematic view of the annotation hierarchy

Modeling and visualizing discussions and argumentations has been used in a few systems before ([Scardamalia and Bereiter, 1993, Edelson et al., 1996, Gordon and Karacapilidis, 1997]). We include here just a small conceptual model, a larger one is being developed right now. Currently, we use a simple hierarchy of appropriate discussion related modeling concepts such as tips, questions and answers as shown in figure 8. Discussions take place within the hyperbook, and are integrated by relating the discussion items (tips, questions, answers, etc.) to the appropriate semantic information units, etc. (figure 5, rel. 1). The annotations for a specific concept for hierarchies are similar to newsgroup threads, but in
contrast to newsgroups, they are not separated from the actual content they discuss, but are integrated in
the hyperbook itself, and connected with the conceptual entities they reference.

5 Conclusion and further work

In this paper we discussed the requirements of constructivist teaching for educational systems based on
the WWW. We proposed a modeling approach based on constructivist teaching and learning functionality
and showed its implementation in a hyperbook used in an introductory course for Java programming.
Further work will concentrate on adapting these conceptual models for the special needs of different
teachers, on an improved integration of material contained anywhere in the WWW, as well as on the issue
of using domain models for semantically structuring large repositories of course materials.

References


Adaptive Interface for Advanced Web-Based Applications. In Proceedings of the Sixth International
Conference on User Modeling, UM97, Sardinia, Italy.

and teaching of science. In Tobin, K., editor, The Practice of Constructivism in Science Education.
Lawrence Erlbaum Associates.

conceptual change: Implications for educational practice. Journal of Research in Science Teaching,
26(9):839–858.

In Wilson, B. G., editor, Constructivist learning environments: Case studies in instructional design,


appeal of the new “ism” (constructivism) in science education. In Tobin, K., editor, The Practice of
Constructivism in Science Education. Lawrence Erlbaum Associates.

framework. In Proceedings of the Sixth International Conference on Artificial Intelligence and Law,
pages 10–18. ACM.


edge Acquisition, 5:199–220.

Formal Ontology in Information Systems. IOS Press.


