Journal of Interactive Learning Research: Special Issue on Intelligent Systems/Tools in Training and Lifelong Learning (2000)
P. Kommers and R. Mizzoguchiv (Eds.), Vol. 11 (3-4), 367-388.

SmexWeb: An Adaptive Web-based Hypermedia Teaching System

Florian Albrecht, Nora Koch, Thomas Tiller

Ludwig-Maximilians-University of Munich
Institute of Computer Science
Oettingenstr 67
80538 München
Germany
{albrechf, kochn, tiller}@informatik.uni-muenchen.de
http://www.pst.informatik.uni-muenchen.de/~kochn

Abstract

A computer learning system that aims at being efficient and gratifying has to adapt itself to the learner's needs. SmexWeb[1] (**S**tudent **m**odelled **ex**ercising on the **Web**) as an adaptive hypermedia system offers advantageous features of the hypermedia paradigm. To take into account spatial and mental abilities of learners various techniques have been developed in the field of adaptive hypermedia. All of them, adaptation of contents as well as all kinds of adaptive navigational support are integrated into SmexWeb. While those techniques move the locus of control towards the system, all action still has to be taken by the user. SmexWeb extends the means of navigation further by being able to take navigation actions for the user based on its student model. By employing a technique, called Passive Navigation, the level of the system's activity can vary subtly and be adjusted to the user's learning preferences.

SmexWeb guarantees easy accessibility, as it is a web-based application that solely requires a standard web-browser. Partially bypassing HTTP as a stateless protocol, SmexWeb enables a higher amount of user-system interactivity than most web-based learning environments. The Java server resembles a classical ITS architecture; the modularity of the framework allows an easy instantiation for future SmexWeb applications. First tests of a SmexWeb application with students have shown that it is an easy-to-use and effective learning environment.

Keywords

Web-based learning, adaptive hypermedia, teaching system, user model, Passive Navigation, Java framework.

1 Introduction

The amount of knowledge available and necessary for everyday tasks is changing more rapidly than ever. According to Rohde (1990), the amount of knowledge mankind deals with doubles

[1] http://pst1.pst.informatik.uni-muenchen.de:8080/

every five years. Thus, acquiring new knowledge is no longer limited to school, university or special training programs. It becomes an increasingly important part of everyday life (Mandl et al., 1997). This implies a broadening of the way we learn and a change in the creation and distribution of learning materials.

During education, we dedicate major parts of the time to acquiring knowledge. Later on, the amount of time available for this task decreases immensely. Finding appropriate information, organising and finally acquiring it has to be efficient, self-regulated and goal-oriented.

In addition, as knowledge tends to become obsolete sooner, it has to be published and distributed faster. Classical media like books do not seem to fulfil this requirement as the process of publication and distribution is cumbersome. Training programs given by experts are a conventional alternative. They solve the problem by avoiding the use of print media, instead presenting current knowledge directly. Nevertheless they are tied to a certain location and the number of participants is restricted.

Life-Long Learning requires an environment that allows inexpensive and easy distribution of knowledge as well as effective and rapid learning.

Publishers need a medium that permits:

- assistance with the pedagogic preparation of learning material,
- rapid publication, updates and distribution, and
- reuse of domain independent features.

Learners require an environment supporting them:

- in effective learning, and
- with an easy access to the learning material.

In this paper we present SmexWeb (**S**tudent **m**odelled **ex**ercising on the **Web)** a framework for implementing teaching systems on the Web. We describe its architecture and components as well as an implemented lesson on the topic "EBNF" (Enhanced Backus-Naur Formalism, a grammar-like technique used for describing the syntax of programming languages).

SmexWeb addresses the requirements for the publishers and learners listed above, as explained in the following. SmexWeb is an *adaptive web-based hypermedia teaching framework*. It implements a higher amount of interaction between the system and the learner than common web-based systems have achieved so far. By widening the classical navigation paradigm of hypermedia systems, using the new concept of Passive Navigation, SmexWeb resembles a teacher/learner situation more closely.

Which advantages does a *web-based* system offer? The World Wide Web is one of the most important and most widely used Internet services. Information identified with a URL is accessible all over the world through common tools called browsers. This means easy distribution and at the same time advantageous maintenance thanks to the local storage of the web application usually on a single computer, the web server.

Why did we choose to develop a *framework*? A framework is a collection of abstract and concrete classes and permits through instantiation the development of teaching applications – we call them SmexWeb applications (Koch, 1998). The framework offers to the publisher the

advantage to concentrate on the subject to be taught and not on implementation details. It allows the reuse of domain independent components. The SmexWeb framework serves as a basis for a future authoring tool.

Why is SmexWeb *adaptive*? The SmexWeb application observes each learner's behaviour and builds a user model for her/him. Based on this user model it dynamically adapts the material to be taught to the learner's characteristics and needs. Presenting appropriate material and individualised guidance to the learner makes the process of acquiring knowledge more effective for her/him. Yet, the user is free to discard guidance of the system. She/he can choose an individual way through the course that seems more appropriate. The level of acceptance of system suggestions in turn provides the latter with valuable information about the user, for instance, her/his self-estimation and way of learning.

Compared to similar systems the number of human-machine interactions that can be observed by the system has been increased in our framework, hence allowing better estimation of the user's needs making learning more efficient. This is accomplished by using a technique we call Direct Interaction. An extra communication channel between the learner's computer and the server is established and used in addition to the stateless Hypertext Transfer Protocol (HTTP) underlying the WWW (Albrecht, 1998). Hence, in addition to all actions a learner takes to request new pages, the system monitors her/his behaviour while interacting with the displayed page. Valuable information, as for instance the way towards a solution of an exercise is available for reasoning about a learner's characteristics. This allows for a better estimation of her/his needs, making learning more efficient.

What are the advantages of a *hypermedia application*? The non-linear structure of a hypertext-based document comprises nodes and links. Nodes – also called pages – contain information. Links connect nodes and give a non-linear structure to the document. The collection of all components of a hypertext document together with the imposed structure is called the hyperspace.

As de Vries, Tiberhien and Guy (1995) state, hypertext documents seem an appropriate way to represent mental models that are usually organised in a non-linear manner. Additionally, the hypertext structure offers the possibility of different paths through the course. By applying adaptive techniques such as adaptive presentation and adaptive navigation support the author can provide the learner with individualised material and suggest appropriate ways of navigating through a course. Adaptive presentation – adjusting the content of the pages to the learner's knowledge and preferences – as well as adaptive navigation support like link hiding, link annotation and link ordering (Brusilovsky, 1996) are supported by the SmexWeb framework. An author defines individual learning sequences by suggesting "ideal" ways through the hyperspace with respect to the system's estimations of a learner.

In addition, SmexWeb introduces a new concept called Passive Navigation. It allows the system to take control over the process of navigation when the learner seems lost or remains inactive, offering her/him some help or guidance (Tiller, 1998).

What kind of learning process do SmexWeb applications support? They support an *active, constructive, cumulative, self-regulated* and *goal-oriented learning process* in which the learners play an important role. Classical psychological theories view learning as something

happening from the outside in – passive reception –, as knowledge transfer from the expert to the novice. Nowadays cognitive theories have reversed this orientation emphasising that learning occurs from the inside out although the importance of the learners environment is not questioned (Shuell, 1992). Instances of our framework SmexWeb are adaptive web-based applications that support learning processes based on the well-known metaphor of problem solving.

Learning with a SmexWeb applications is:

- *Active* as the learner must carry out cognitive operations (learning by doing).
- *Constructive* in the sense that it helps every learner to create her/his own knowledge structures. New information is perceived and interpreted in a unique manner based on the learner's prior knowledge and other personal factors.
- *Cumulative*, because it builds upon and is influenced by the learner's prior knowledge, which is registered in the user model.
- *Self-regulated* as the learner determines the duration and frequency of the sessions. They are free to decide which link to choose next, although they are assisted by the system with some guidance and help.
- *Goal-oriented* for the learner. The application presents her/him clear goals to achieve in each session within the context of the general goal to acquire knowledge about certain topics. The learner can then establish her/his goal for the session.

This paper is structured as follows: In Section 2, the SmexWeb architecture is presented explaining new features like increased interaction and Passive Navigation. A summary of all actions necessary for building a SmexWeb application are given in Section 3 where the EBNF example application is presented. We describe a test of this application. Section 4 briefly compares SmexWeb to other web-based adaptive systems. Finally, Section 5 outlines some conclusions future work.

2 The Architecture of SmexWeb

The following chapter describes the components that make up the SmexWeb framework as shown in Figure 1. These are the web client, a HTTP server and the SmexWeb server. The focus will be on the SmexWeb server, a collection of reusable abstract and concrete classes written in the Java programming language.

OVERVIEW

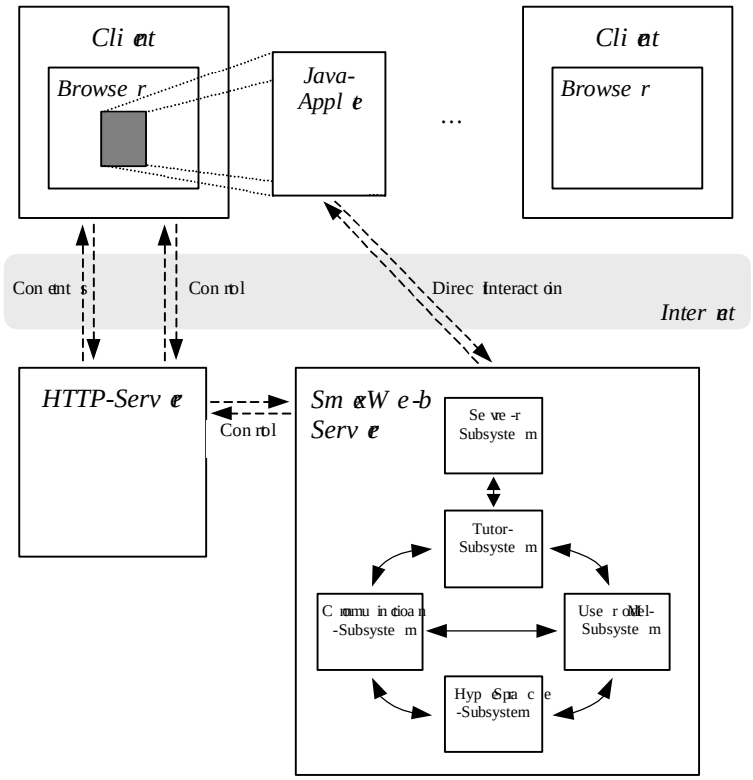


Figure 1: Architecture of SmexWeb

The SmexWeb architecture resembles a classical client/server concept that is built upon the WWW. One of our aims was to reuse as many common technologies and products as possible. We use a standard web server to transfer all the content material to the learner and to pass data on to the SmexWeb server application and back to the client. Identification of learners is accomplished by using HTTP access authentication. Most information exchanged between client (learner) and server (SmexWeb) via the Internet is transported using HTTP, which is the native protocol of the WWW. The content presented to the learner is composed of standard HTML pages, which may contain any media type a web browser is capable of displaying. The adaptation of the pages presented to the learner is performed by JavaScript programs embedded in the HTML code. Using standard products and technologies has several advantages: functionality and efficiency of common products are well proven, greater platform independence can be reached since all those products are equally available for different systems.

A learner can utilise the system from any computer that has Internet access by pointing a standard web browser to a specific URL. No other software has to be installed on the client computer to run SmexWeb. The learner can start working with a SmexWeb application immediately.

A typical learner's request is processed as follows:

- The HTTP server passes the request on to the SmexWeb server.
- SmexWeb creates a component called Tutor for every learner logged on to the system. The Tutor is responsible for keeping all necessary information about the learner and for processing her/his requests.
- The tutor analyses the information package sent by the learner's browser incorporating relevant information about the learner into the user model. This is done by a set of acquisition rules provided by the HyperSpace as explained below.
- Once the user model is up to date, the Communication Subsystem generates a small JavaScript program representing control information that is necessary for constructing an answer to the request. The answer is based on the data contained in the HyperSpace and UserModel subsystems.
- This program is sent back to the browser via the HTTP server.
- The client browser executes the JavaScript that retrieves all the media necessary for assembling the pages of information and displays them according to the user model.

In this way the workload is distributed between server and client side. Furthermore, as the content adaptation takes place on the client side, the amount of data transferred over the network can be minimised. For the learner this means faster response time of the system, which is an important factor in preventing loss of motivation.

THE CORE OF SMEXWEB

The subsystems of the SmexWeb server framework correspond to the typical Intelligent Teaching Systems (ITS) components as described in (Beck, Stern and Haugsjaa, 1996) whereas the Domain Knowledge and Expert Model components are substituted by the HyperSpace component. Those modules of the framework that have to be instantiated for a concrete application will be outlined.

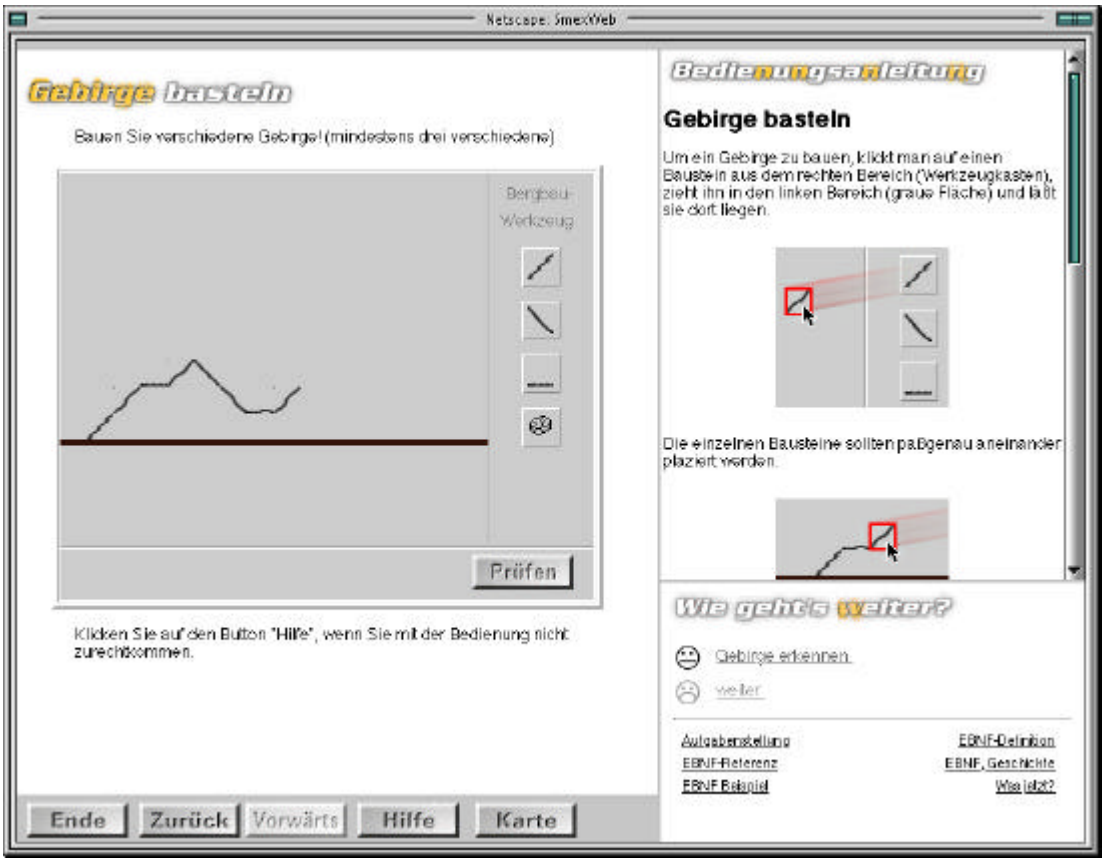
The Tutor Subsystem

The Tutor Subsystem is the heart of the server application. Following the metaphor of a private teacher a Tutor object keeps track of all information bound to a single learner. Several users may use one SmexWeb application at the same time, each one with her/his own Tutor, the Tutors working in parallel. The Tutor observes the learner working with the material, builds a model of her/his preferences and capabilities, and gives her/him assignments and answers to questions according to this model.

For an accurate representation of the learner's characteristics, our framework allows a higher degree of interaction and observation than common web-based systems. The WWW paradigm only transmits data when a user follows a link. This allows for an easy implementation of fill-out questionnaire tests in a learning environment. Many web-based teaching environments work in this way. (Kay and Kummerfield, 1994; Weber and Specht 1997; Nakabayashi et al. 1997) A private teacher however will usually judge a learner's performance not only by the result of a test but also by how she/he achieves the solution. The teacher might want to give the learner some hints along the way.

At this point we extended the WWW paradigm by an additional communication channel between client and server to allow for a higher degree of interaction. Java applets running in the learner's web-browser communicate directly with the SmexWeb server as shown in Figure 1 (Direct Interaction). This extension gives the author of a teaching system the ability to collect more information about the learner, which in turn enables better adaptation of the material to the learner's needs. Figure 2 shows an interactive exercise in the EBNF application. The left part of the window contains an applet implementing the exercise. Not only final solutions to the exercise but also important events are transmitted to the SmexWeb server. For a more detailed description of the user interface see Section 3.

An author of a teaching system can use the Tutor Subsystem directly without modification since no domain dependent information is contained.



The UserModel Subsystem

The user model maintains the assumptions the system has about the user (Kobsa and Pohl, 1995). Topics to be covered when building an adaptive system are: what information about the

Figure 2: An interactive exercise in the EBNF example application

user is to be modelled, how is the model organised and how is the information about the user acquired.

Several teaching systems in different domains expose a partition of the user model into three sections (Benyon and Murray, 1993; Murphy and McTear, 1997; Kay and Kummerfield, 1994). The most common way of organising data is the division into learner's domain knowledge, cognitive characteristics such as preferred media and learning strategies and the user's profile including interests and knowledge in other domains. The latter two are likely to remain constant for a longer period and may be reused in teaching different subjects.

The SmexWeb framework supports the author in creating a user model for a SmexWeb Application by providing different modules: the user model and a number of sub-models. The user model organises access and manipulation of the sub-models. It is the interface to the rest of the system and hides organisation of the data. The current version of the framework contains three sub-models as mentioned above. More details about these sub-models are given in Section 3. The sub-models offer an easy, intuitive way for rapid application development, by providing a data structure based on key-value pairs. Yet, the author is free to create sub-models of any kind and complexity reusing only some parts of the UserModel component.

Different values of a user model may depend on each other. To maintain consistency within the model SmexWeb provides a mechanism of so-called consistency rules. Interdependent parts of the model may be connected by those rules and constraints may be stated. How these constraints are formulated, as simple conditions or as more complex calculations, is up to the author.

Acquiring the knowledge about a user is implemented by so-called acquisition rules based on condition-action pairs. Any screened interactions as well as all the information about a user are available to formulate conditions and subsequent actions. By instantiating the condition-action pairs, the author has a straightforward way to implement her/his strategies of estimating learner's characteristics in a procedural, hence intuitive way.

The HyperSpace Subsystem

Using the non-linear structure of hypertext documents for the representation of knowledge bares a variety of features, supporting teaching and learning in a positive way.

As Streitz (1990) points out cognitive models of knowledge usually are structured in a non-linear way. Using a non-linear representation helps conveying not only the knowledge but also its structure to a learner. It avoids the cognitive overhead of linearisation of knowledge by an author and the reconstruction of a mental model by the learner. The learner will less likely build faulty cognitive models.

Some issues have to be considered when developing adaptive hypertext systems:

- separation of structure and content,
- locus of control over navigation, and
- consistency of the presented material.

The SmexWeb framework addresses these issues as follows:

SmexWeb clearly separates the structure of a hypertext document from the physical pages presented to the learner. This separation is suggested by many well known hypertext reference models as for instance the Dexter Hypertext Reference Model (Halasz, 1994). The author has the freedom to take different steps in the creation process of a course untied from each other:

The representation of the mental model and the suggestion of individual ways through it are a matter of the courseware structure. The author builds a graph of links and nodes by instantiating the respective classes of the framework. She/he declares the importance of the links and nodes for the individual learner. Different link representations and annotations may be specified dependent upon different user model states.

The presentation and adaptation of material is covered by the step of creating pages. Each page in the hypertext structure is linked to one physical page, represented by a downloadable HTML file. The idea of writing an adaptive page is based on page fragments as suggested by Kay and Kummerfield (1994). SmexWeb widens this concept by allowing any kind of displayable item. An author can explain concepts in a variety of ways and assign each fragment to certain users. According to the state of the user model, the most appropriate alternatives are presented to the user. Examples are alternative versions of a single word or a whole video that

might only be presented to learners preferring this kind of media. Adaptation of the HTML file at run-time is done by JavaScript programs that show and/or hide certain parts of the page.

By using these standard technologies for adaptation, parts common for all different learners only have to be created and maintained once. A further advantage is the ease-of-reuse of existing learning material, given that it is written in HTML.

So far, the user mostly controls the way through the course material and the pace of learning. This ensures an active, constructive, self-regulated and goal-oriented process of knowledge acquisition. The author supports the learner by suggesting links and annotating them individually. Yet, sometimes it is necessary, that the system takes over more control (Vassileva and Wasson, 1996). The conventional hypertext paradigm does not allow the system to lead a learner to a different page without her/him taking the action of navigating there.

The SmexWeb framework widens this paradigm by implementing a new way of navigation, so-called Passive Navigation. The idea is *that the system has the possibility to take control over navigation if necessary*. If the assumptions about a user and her/his inactivity indicate a different page than the currently displayed one as more appropriate the system navigates the user to that page. The SmexWeb framework allows the author to state target nodes, where learners might be lead to under certain circumstances. This way a more natural teacher/learner situation can be simulated. The user's feeling of being lost and subsequent frustration and demotivation are avoided.

Finally, *the SmexWeb framework addresses an intricate topic in adaptive hypertext systems: consistency*. A user may want to walk back a path she/he had followed for a couple of pages. The material on a page is adapted to the user according to the user model state at display time. Navigating back to a previously read page might become confusing, as the user model might have changed over time and the page might look completely different from what the learner expects. The SmexWeb framework provides a built in mechanism to avoid frustration coming up in this situation: it maintains a personal history including the already visited pages and user model states. When a learner revisits a page using the back button, the page automatically is displayed as it was when visited the last time.

The Communication Subsystem

The task of the Communication Subsystem is to produce the answer to a given request from the client. Information from UserModel and HyperSpace is collected and integrated to form the answer. As mentioned above the SmexWeb generates control information, while the client browser has the task to assemble content pages.

The author of a concrete SmexWeb application may use the Communication Subsystem without modification as it provides a default handling. However, all aspects that might be subject to desired change can be modified. The Communication Subsystem contains all heuristics to translate the information for adaptation provided by the HyperSpace module into concrete adaptive navigation techniques.

3 Building a SmexWeb Application

This section will give an impression of the process of building a SmexWeb application. Each step is outlined and illustrated by an existing application. Finally, a test of the introduced example is described.

Building a SmexWeb application is a multiple-phase project. The steps are:

- classifying potential users,
- designing the user model,
- structuring the hyperspace,
- authoring pages, and
- designing the user interface.

The first SmexWeb application was developed for students at the Ludwig-Maximilians-University in Munich. The application offers the possibility to practice EBNF, a formal grammar used for programming language description. The availability of a heterogeneous group of motivated students determined the choice of the topic for this SmexWeb application. Although the domain in this case is computer science, the framework is generic enough to create courses in any other domain.

Classifying potential users

To adapt any presented material close to individual needs, the learner must be modelled as accurate as possible. Classifying potential users before designing the user model might avoid modelling too generic characteristics. Consequently, without making the model more complex a more precise representation of the learner's characteristics can be achieved. The author can keep the user model small, the system works more efficient and the learner's needs are met better.

EBNF is taught in an introductory computer science course. The course is offered to students of various subjects whose minor is computer science. Students from the first up to the eighth semester as well as elderly people attend the course. The SmexWeb application on EBNF is intended for this heterogeneous group. Neither experience in working with computers and interactive systems nor the ability of understanding and applying abstract formalisms could be taken as prerequisite.

Designing the user model

Characteristics to be modelled are determined based on the given decisions about the topic and the potential group of users. The current user model for the EBNF course comprises three sub-models as introduced in Section 2, called domain, navigation and individual model. Values of the domain model represent the learner's knowledge about the topic of the course. The most important attribute of the navigation model captures the learner's navigation experience. The individual model represents learning preferences, for instance with brief or extended explanations, more abstract or more concrete descriptions. The initial values of all sub-models are assigned on basis of a learner's answers to interview questions. From there on, values will

be changed dynamically according to the learner's actions while navigating or solving an exercise.

In many cases the creation of the user model is a creative and non-deterministic process relying heavily on the pedagogical experience of the author and her/his knowledge about existing psychological models. The planned SmexWeb authoring tool will provide the functionality to be flexible enough to build user models supporting different pedagogical strategies. The current SmexWeb framework supports a rapid transformation of pedagogic knowledge into a working system in two ways: The partition of the user model suggested above and easy reuse of already proven and domain independent parts of the model.

In our case, the uncertainty about how experienced the students are with computers and interactive systems makes those two attributes indispensable. The topic of EBNF requires the representation of the learner's cognitive abilities to formalise and to think in abstract terms. As students might not know about EBNF when they use the SmexWeb course to practice, pre-knowledge in the domain is taken into account. Background knowledge, derived for instance from the learner's major subject, is included into the user model to support learning by drawing analogies.

Structuring the hyperspace

The hypertext structure reflects the non-linear representation of the domain knowledge and provides different ways through it. The author has to use pedagogical as well as her/his domain knowledge to structure the document. A first coarse outline and a stepwise refinement make this process easier. In the final version, different individual ways to navigate through the hypertext are stated based on the user model.

Figure 3 shows the coarse architecture of the EBNF course. After an initial interview phase where the system builds first assumptions about the student, an exercise is presented to the learner. In the next two steps, the student interactively solves the exercise with the support of the system. The system offers different ways of reaching this goal, depending on the estimated cognitive abilities of the user. Finally, the acquired knowledge and skills are applied by solving a similar exercise without support. At any time, the student may request pages containing context sensitive help and explanation of the domain concepts. Under certain circumstances, the system might present these pages to the user by means of Passive Navigation. As an alternative to the interactive way, a more difficult exercise is offered to learners who already know EBNF well.

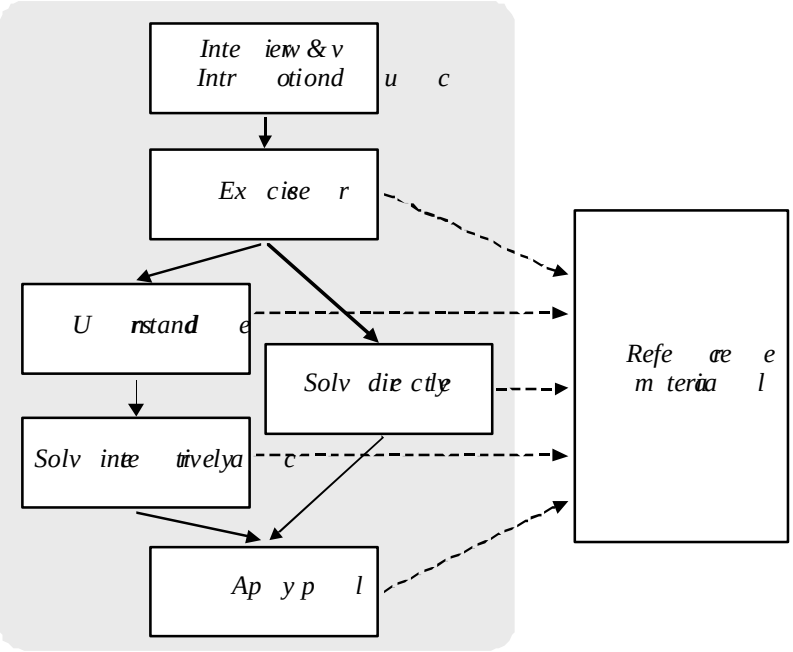


Figure 3: Structure of the hyperspace in the EBNF application

Authoring pages

Besides constructing the hyperspace the author writes pages containing the adaptable material. As described in the previous section, the author provides different alternative presentations for concepts explained to the learner. The decision about how small differences between page fragments are is left to the author. By declaring the user model state at which a fragment is presented to a learner the author has a straightforward way of individualising explanations.

For instance, Figures 4 and 5 shown below depict the same page presenting the initial exercise to the learner. A formal way of explaining the problem is chosen in Figure 4, while a descriptive formulation appears in Figure 5. Dependent on the cognitive preferences and abilities of a student represented in the user model, the appropriate form of presentation is provided.

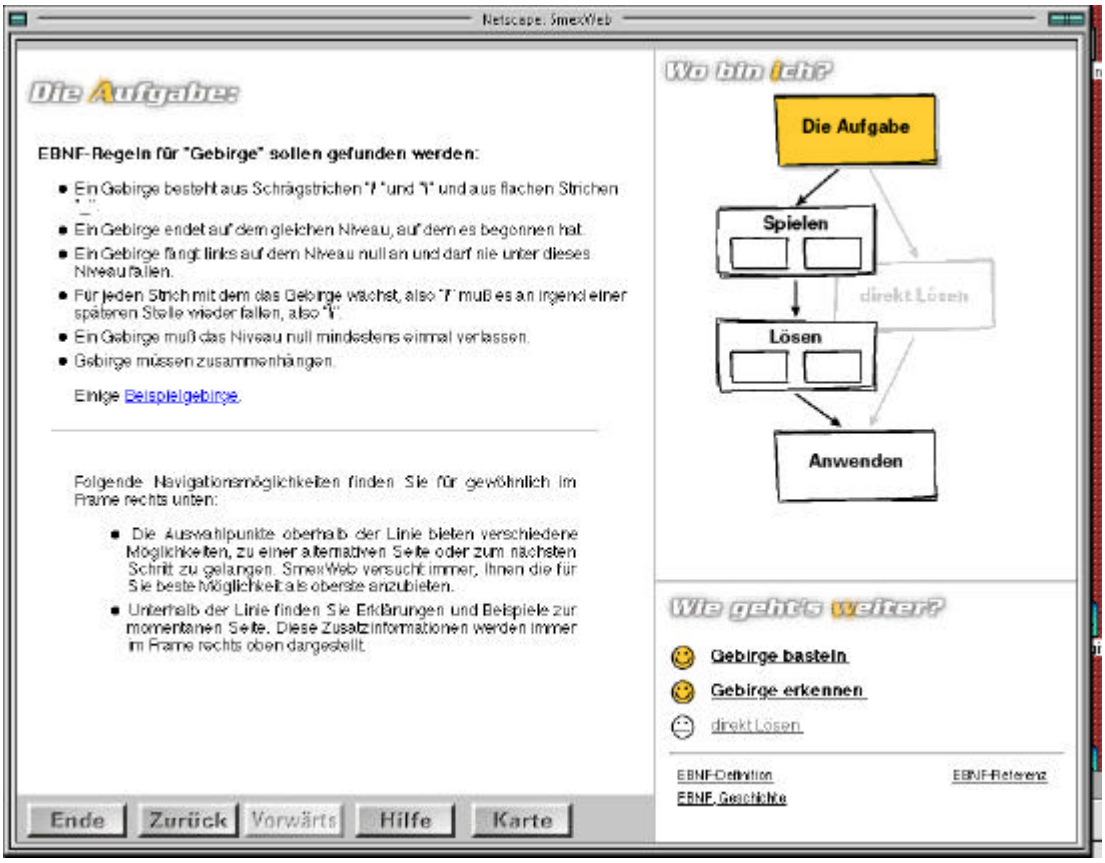


Figure 4: Exercise page in SmexWeb-EBNF using formal description.

Designing the user interface

The SmexWeb framework gives the author a great freedom of designing the user interface based on the notion of logical windows. These windows fulfil different tasks and present different kinds of material. Standard web browsers offer the possibility of opening new windows as well as dividing one window into different areas, so-called frames. SmexWeb supports both variants.

To allow a rapid instantiation the framework offers a standard window partition using frame technology. Figures 2, 4 and 5 show this particular division with the following underlying concepts:

- The work area (upper left) contains the pages with the material the student currently is working on, as for instance interactive pages or important explanations.
- The help area (upper right) presents pages related to the current page in the work area, as context sensitive help or a site map.
- The navigation area (lower right) provides the user with the navigation facilities in an ordered and annotated way. In this SmexWeb application the links are ordered according to the estimated individual importance and annotated by icons and greyscale levels.
- The button bar (lower left) comprises navigation facilities that are always available to the learner as for example the back button, displaying the previously presented page when used.

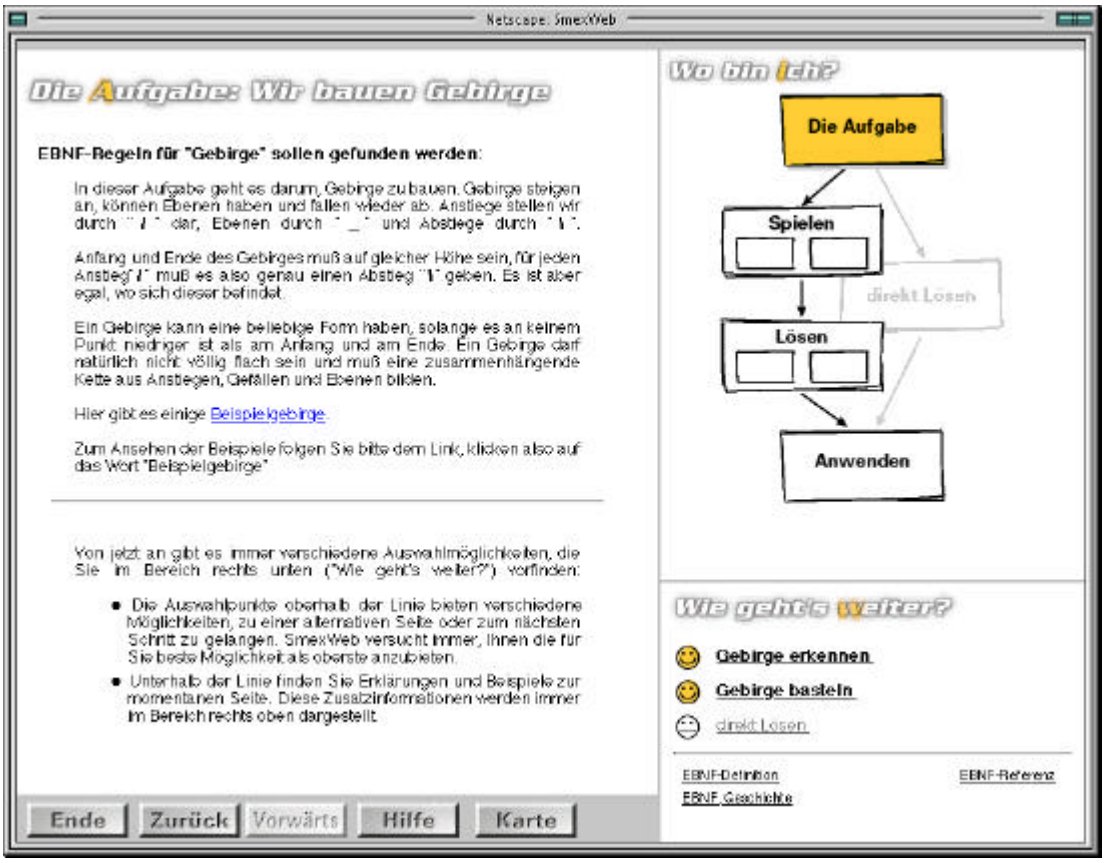


Figure 5: Exercise page in SmexWeb-EBNF using narrative description.

Testing the SmexWeb application

To prove our concept we carried out a small test of the SmexWeb EBNF application. The focus was on having a real world situation without any specific guidance or introduction to the system. Sixteen people at an age ranging from 22 to 66 used the system for an average of about 40 minutes. The system asked them to answer a few general and topic-dependent introductory questions to get initial values for the user model, but they were not explicitly told that the system would try to adapt to their needs. All students were immediately able to concentrate on the domain without any further explanations on how to use the SmexWeb application. This is an important prerequisite for enabling effective learning with computers.

Subjects were highly motivated during their use of the system and most of them gave very positive comments in a post-questionnaire. From this first test we can conclude that the system works in a very efficient and stable way. System response time was very short compared to network latency and data transfer time.

4 Related Work

Research on adaptive interactive systems is a very active field, especially in the area of adaptive teaching systems. Recently many of these systems have been deployed on the Web. In this section, we outline a few aspects that distinguish SmexWeb from other web-based educational systems.

ELM-ART II (Weber and Specht, 1997) is an example of a web-based tutoring system that evolved from being a stand-alone ITS. It uses Common Lisp to dynamically generate all the HTML pages based on the student model and the stored domain knowledge. The learner's behaviour is observed and data is collected to form an individual learning history. This so-called episodic learner model is quite complex with the advantage of being much more precise than usual models. All data is gained through question and answer tests via HTML forms.

In the SmexWeb framework we chose a simple user model focussing on ease of reuse of model parts in different domains. However, SmexWeb not only acquires information about the learner through answers to questions, it additionally registers and interprets other events such as mouse clicks and selected navigation paths. All this data can be used to adjust the user model.

InterBook is a tool for authoring and delivering adaptive electronic textbooks on the WWW (Brusilovsky, Eklund and Schwarz, 1998). Courses developed with InterBook have a textbook structure, i.e., they require a hierarchical organisation of the learning material in lessons, sections, subsections and terminal pages. It is based on the same approach as and on the experience gained with ELM-ART. Extensive adaptive navigation support is provided. It uses an extended traffic light metaphor shown as coloured bullets combined with different font styles for link annotation. InterBook also adds a check mark for already visited nodes. Furthermore it includes a frame called navigation bar that provides the student with a hierarchy of surrounding nodes in the textbook structure. While these links can be very useful for the learner all navigation facilities take away screen real estate and put a cognitive load onto the learner. In a test conducted by the authors, students were familiar with the user interface of a textbook after a two-hour introductory session.

When developing our EBNF system we concentrated on a simple interface where the content of the course is always the main focus. Consequently, the work area takes the biggest part of the screen. SmexWeb in its standard implementation uses three different icons in connection with a text fade-out metaphor for annotating links. Irrelevant links may be hidden. While not as supportive as the link annotation in InterBook for frequent users, learners were able to work with SmexWeb instantly. InterBook's mode of link annotation could however be easily implemented with the SmexWeb framework. While hierarchical structuring of the content might be natural for authors of learning material SmexWeb specifically supports arbitrary graph structures - including hierarchical ones - and does not use the book analogy.

In contrast to the systems described so far, Vassileva´s (1997) authoring system DCG (Dynamic Course Generation) presents an architecture for an open corpus environment. It can use any HTML document as teaching material with the advantage of fast generation of a learning curriculum. The tool generates individual course material according to the student goals and previous knowledge based on a probabilistic student model. Since the concept

structure is separated from the teaching material, authoring consists of two parts: creating or modifying a concept structure for a given domain and providing links from each concept/relation to appropriate HTML files.

SmexWeb is a framework with the goal of generating a closed corpus of specifically prepared material. In our approach, preparation of the learning material certainly causes more work, but it leaves the author the complete control over all pedagogical aspects of the course.

5 Conclusions and Future Work

The first test of the EBNF course has confirmed that a SmexWeb application meets the learner's requirements of Life–Long Learning: effective knowledge acquisition and easy accessibility. Even students with hardly any computer experience were able to use the system without introduction. The system's performance was reliable, integrated error recovery mechanisms worked well.

Currently, the SmexWeb framework does not cover the important topic of collaboration between different learners and between authors and learners respectively. The future integration of an interactive discussion forum will allow easy communication.

Creating the SmexWeb application on EBNF has shown that the framework takes away the burden of knowing many technical details of web-based interactive applications from the author. Yet, knowledge in computer science and programming skills are still necessary to instantiate our framework. We will address this issue by building an intuitive tool on top of the framework, providing editors for the different phases in the process of creating a SmexWeb application. A graphical editor based on a hypertext meta model like the one introduced by Duval, Olivie and Scherbakov (1995) will ease the creation of large hypertext documents.

Accompanying the above mentioned tool, a method for designing adaptive hypermedia applications is currently being developed (Koch, 1998).

A second SmexWeb application in the field of biology, which is currently being built, and further tests will give more insight into possible extensions and improvements of the framework itself. Our goal is to apply SmexWeb as a learning and exercising system also in non-academic areas. We think that SmexWeb could be an ideal training system for new procedures and methods in industrial and commercial organisations.

References

Albrecht F. (1998). SmexWeb: Ein adaptives web-basiertes Übungssystem. Diplomarbeit, Ludwig-Maximilians-Universität München. http://pst1.pst.informatik.uni-muenchen.de:8080/DA/

Beck J. & Stern M. & Haugsjaa E. (1996). Applications of AI in Education. ACM Crossroads, 3 (1). http://www.acm.org/crossroads/xrds3-1/aied.html.

Benyon D. & Murray D. (1993). Applying User Modeling to Human-Computer Interaction Design. Artificial Intelligence Review, 7, 199-225, Kluwer Academic Publishers.

Brusilovsky P. (1996). Adaptive Hypermedia: An Attempt to Analyze and Generalize. Proceedings of Multimedia, Hypermedia and Virtual Reality. P. Brusilovsky, P. Kommers and N. Streitz (Eds.). Lecture Notes in Computer Science 1077, 288-304, Springer Verlag.

Brusilovsky P., & Eklund, J. (1998). A study of User Model Based Link Annotation in Educational Hypermedia. Journal of Universal Computer Science, 4 (4), 429-448, Springer Science Online.

Duval E. & Olivie H. & Scherbakov N. (1995). Contained Hypermedia. Journal of Universal Computer Science, 1 (10), 687-705, Springer Verlag.

Halasz F. & Schwartz M. (1994). The Dexter Hypertext. Communications of the ACM, 37 (2).

Kay J. & Kummerfield R. J. (1994). An Individualised Course for the C Programming Language. Proceedings of the Second International WWW Conference '94, Mosaic and the Web.

Kobsa, A. & Pohl W. (1995). The User Modeling Shell System BGP-MS. User Modeling and User-Adapted Interaction, 4 (2), 59-106.

Koch N. (1998). Towards a Methodology for Adaptive Hypermedia Systems Development. Proceedings of the Sixth Workshop ABIS-98: Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen, U. Timm, and M. Roessel (Eds.), 41-52, FORWISS.

Mandl H. & Reinmann-Rothmeier G. (1997). Zukunft Cyberspace? Europas Weg in die globale Informationsgesellschaft. Presentation at the Europian Colloquium in Regensburg, Germany.

McTear M. (1993). User modelling for Adaptive Computer Systems: A Survey of Recent Developments. Artificial Intelligence Review, 7, 157-184, Kluwer Academic Publishers.

Nakabayashi K. et al. (1997). Architecture of an Intelligent Tutoring System on the WWW. Proceedings of the Eighth World Conference of the AIED Society.

Murphy M. & McTear M. (1997). Learner Modeling for Intelligent CALL. Proceedings of the Sixth International Conference of User Modeling: UM´97, Jameson A., Paris C. and Tasso C. (Eds.), 301-312, Springer Verlag.

Rohde S. (1990). Wie uns das Wissen schafft. Magazin der Süddeutschen Zeitung, 30, 11-17.

Shuell T. (1992). Designing Instructional Computing Systems for Meaningful Learning. Adaptive Learning Environments, Foundations and Frontiers, M. Jones and P. Winne (Eds.), 19-53, Springer-Verlag.

SmexWeb: Student Modelling Exercising on the Web (1998). http://pst1.pst.informatik.uni-muenchen.de:8080/

Streitz N. (1990). Hypertext: Ein innovatives Medium zur Kommunikation von Wissen. Hypertext und Hypermedia, Gloor P. and Streitz N. (Eds.), Springer-Verlag.

Tiller T. (1998). Eine Hypertext-Struktur für abgeschlossene adaptive Systeme. Diplomarbeit, Ludwig-Maximilians-Universität München. http://pst1.pst.informatik.uni-muenchen.de:8080/DA/

Vassileva J. & Wasson B. (1996). Instructional Planning Approaches: from Tutoring towards Free Learning. Proeedings of EuroAIED'96, 1-8.

Vassileva, J. (1997). Dynamic Course Generation on the WWW. Proceedings of Workshop Intelligent Educational Systems on the World Wide Web in AI-ED´97: Eighth World Conference on Artificial Intelligence in Education.

de Vries E., Tiberhien A. & Guy P. (1995). Learning Processes and Knowledge Representation in the Design of Educational Hypermedia. Proceedings of Hypermedia Design 95.

Weber G. & Specht M. (1997). User Modeling and Adaptive Navigation Support in WWW-Based Tutoring Systems. Proceedings of the Sixth International Conference of User Modeling, UM´97, Jameson A., Paris C. and Tasso C. (Eds.), 289-300, Springer Verlag.