In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.), *User Modeling: Proceedings of the Sixth International Conference, UM97*. Vienna, New York: Springer Wien New York. © CISM, 1997. Available on–line from http://um.org.

User as Student: Towards an Adaptive Interface for Advanced Web-Based Applications

Peter Brusilovsky[1] and Elmar Schwarz[2]

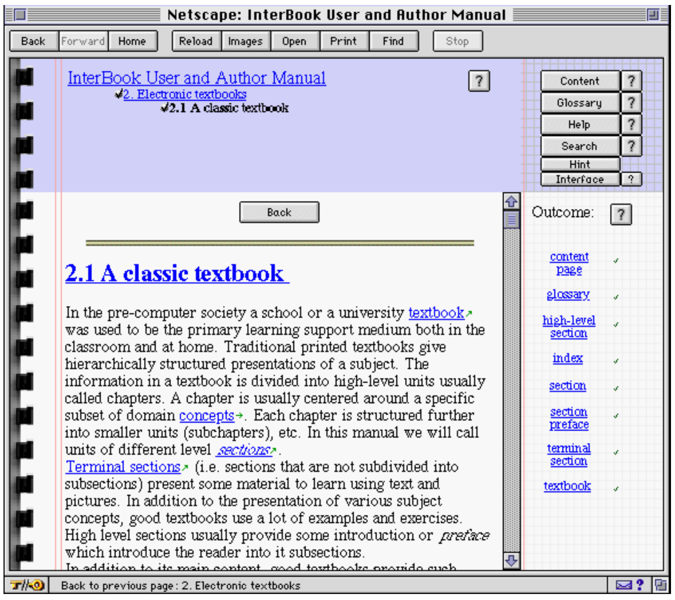
[1] Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, U.S.A.
[2] Department of Psychology, Carnegie Mellon University, Pittsburgh, PA, U.S.A.

Abstract. This paper discusses the problems of developing adaptive self-explaining interfaces for advanced World-Wide Web (WWW) applications. Two kinds of adaptation are considered: incremental learning and incremental interfaces. The key problem for these kinds of adaptation is to decide which interface features should be explained or enabled next. We analyze possible ways to implement incremental learning and incremental interfaces on the WWW and suggest a "user as student" approach. With this approach, the order of learning or enabling of interface features is determined by adaptive sequencing, a popular intelligent tutoring technology, which is based on the pedagogical model of the interface and user knowledge about it. We describe in detail how this approach was implemented in the InterBook system, a shell for developing Web-based adaptive electronic textbooks.

1 Introduction

Current advanced World Wide Web-based applications are more than networks of static hypertext pages. They offer a rather complex interface with a number of different windows, subwindows (frames), forms, and buttons. What is often missing is adaptivity and adaptability. Users of these applications with different abilities, Web experience, knowledge, and background get the same pages in the same context. The class of users who need adaptivity really urgently comprises millions of Web newcomers. These users, who have almost no Web experience (and often no general computer experience), just cannot deal with complex interfaces of advanced Web-based applications. The problem of how to help novice users of advanced WWW applications is the first one that has to be solved on the way to adaptive Web-based systems, and this paper is centered around this problem.

A good example of advanced Web-based applications is found in *interbooks*. These are adaptive electronic textbooks (AET) that are developed with InterBook, a tool for the authoring and delivery of Web-based AETs (see http://www.contrib.andrew.cmu.edu/plb/InterBook.html; and Brusilovsky et al., 1996). AETs served on the Web by InterBook have a powerful and complicated interface. InterBook uses several separate windows, further subdivided into frames, to provide the user with useful tools, orientation support, navigation support, and other interface features that are known to be useful from the research literature (Figure 1). InterBook is being used to develop AETs for courses offered at Carnegie Mellon University. Communicating with people involved in teaching various courses, we have found that the InterBook interface is too complex for many users. Some interface features, such as the separate table of contents, were misunderstood. Such helpful features of InterBook as the search interface or prerequisite help have never been used.



2.1 A classic textbook

In the pre-computer society a school or a university textbook was used to be the primary learning support medium both in the classroom and at home. Traditional printed textbooks give hierarchically structured presentations of a subject. The information in a textbook is divided into high-level units usually called chapters. A chapter is usually centered around a specific subset of domain concepts. Each chapter is structured further into smaller units (subchapters), etc. In this manual we will call units of different level *sections*.
Terminal sections (i.e. sections that are not subdivided into subsections) present some material to learn using text and pictures. In addition to the presentation of various subject concepts, good textbooks use a lot of examples and exercises. High level sections usually provide some introduction or *preface* which introduce the reader into it subsections.
In addition to its main content, good textbooks provide such

Figure 1. This picture shows a complex interface of InterBook. All features are switched on. The complexity of the interface often confuses novices. At the top right corner you can find the “Hint” and the “Interface” button, which are used for interface customization.

We think that InterBook provides a good case for investigating the problem of novice users working with complicated interfaces on the Web. The WWW context adds new dimensions to this classic problem of human-computer interaction. Generally speaking, there is a whole tree of computer-supported solutions (not mutually exclusive) for this problem. Either the user has to adapt to the complex interface by learning it, or the system has to adapt to the user’s level of knowledge and skills. In turn, an interface could be learned in advance (comprehensively) using some kind of on-line tutorial; or at work (incrementally) using some kind of on-line help. Experimental research shows that some users prefer the comprehensive and others the incremental way of learning (Fischer, 1988; van der Veer, 1990). On-line help could be static (i.e., non-adaptive) or adaptive.

We have already tried two non-adaptive options within this taxonomy in InterBook. From the very beginning, InterBook was available together with the “InterBook manual”, a combination of on-line tutorial and on-line help. Unfortunately, very few users have ever tried to read this manual. Either users are too eager to start real work as soon as possible in the WWW context, or the reading of the Web-based manual was a complicated task in itself. This paper presents our efforts to investigate two adaptive options: incremental learning with adaptive help; and an adaptive interface.

2 A Review of Relevant Work

2.1 Adaptive Help Systems for Incremental Learning

The problem of incremental learning of complex interfaces has traditionally been addressed by the research on adaptive help systems (more often called *intelligent help systems*, IHSs). The goal of intelligent help systems is to provide personalized help to a user working with a complex interface by diagnosing errors and suboptimal user behavior, identifying missing pieces of knowledge about the interface, and providing on-demand incremental learning to extend the user's knowledge (for a review see Breuker, 1990; Wasson and Akselsen, 1992).

The area of intelligent help systems has been well investigated. The first wave of research on intelligent help was initiated when UNIX systems, with their complicated interface, were distributed widely in universities and came to the workplaces of many computer-naive users. Due to this fact, almost all early research on intelligent help systems was focused on UNIX and its utilities (Breuker, 1990; Jones et al., 1988; Matthews and Nolan, 1985; Nessen, 1989; Wilensky et al., 1984; Wolz et al., 1989). The appearance of "friendly" WIMP interfaces created a pause in IHS research, but in just a few years even these interfaces had reached the level of complexity where intelligent help is really important. The current second wave of research on IHSs investigates useful ways of providing intelligent help in modern application systems (Encarnação, 1995; Fox et al., 1993). Probably, a third wave will be created by advanced WWW applications.

Traditionally, IHSs are divided into two classes: *active* and *passive* help systems. In a passive help system, it is the user who initiates the next help session by asking for help. An active help system initiates the help session itself. "Passive-active" and "static-adaptive" are two different dimensions of classification. A well-known example of active but non-adaptive help is "did you know" (DYK) help, which offers users random pieces of knowledge (called *hints*) during their work. A number of modern applications (like Microsoft Word) usually suggest DYK help at the beginning of a session.

Adaptive passive help systems support incremental learning by suggesting the next piece of knowledge to be learned by the user when help is requested. The main problem for these systems is how to decide what to say. The suggested piece of knowledge has to be both new and relevant to the user's current goal (otherwise the user will not be interested in learning it!). To determine what is new and relevant, IHSs track the user's goals and the user's knowledge about the interface. This information about the user is stored in the user model (Chin, 1989; Nessen, 1989; Winkels, 1990). The user model is often initialized through a short interview with a user and then kept updated through automatic user modeling. IHS researchers have investigated a number of effective techniques of automatic user modeling. Most of them are variants of two basic technologies which were tried in the very first IHS projects (Matthews and Nolan, 1985; Zissos and Witten, 1985): (1) tracking the user's actions to understand which commands and concepts are known to the user and which are not and (2) using task models to deduce the goal of the user. The first technology is reasonably simple—the system just records all used commands and parameters, assuming that "used means known". The second technology is much more complicated; it is based on plan recognition (Goodman and Litman, 1992) and advanced domain knowledge representation in a "goal-plan-action" tree or some other form of task model (Hoppe, 1993). To identify missing pieces of knowledge, the system (1) infers the user's goal from an observed sequence of commands, (2) tries to find a more efficient sequence of commands to achieve this goal, (3) identifies knowledge elements required to build this more efficient sequence but not required to make the original (less efficient) sequence. Presumably, the lack of this knowledge prevented the user from applying the more efficient sequence. These knowledge elements are declared to be unknown (unless they were recorded as known on the basis of the simpler technology) and become the best candidates for knowledge extension on demand.

Adaptive active help systems make all decisions that adaptive passive help systems make plus one more decision: when to interrupt the user's work and suggest help. The problem of when to interrupt is older than IHSs themselves. This problem is known in the area of intelligent tutoring systems (ITSs) as the problem of *coaching* (Burton and Brown, 1979), and here IHSs apply the ideas developed by earlier ITSs (Breuker, 1988; Fischer, 1988). The secret of good coaching is not to interrupt the user's work in each situation when the user makes an error or demonstrated suboptimal behavior and the system has something to say about it. A good coach will interrupt the user only if the work situation is relevant for correcting the user or suggesting a new piece of knowledge. Usually, a set of heuristic rules (mostly domain-dependent) is used to determine whether the situation is relevant.

2.2 Adaptive Interfaces

Interface adaptation to users with different levels of knowledge about interface features is not a mainstream direction of research on adaptive interfaces (Dieterich et al., 1993). The only relevant technology developed so far is that of incremental interfaces. The idea of incremental interfaces is the following one: A novice user starts with some subset of a complex interface, and then more advanced interface features are enabled incrementally as soon as the user needs them and is ready to use them. Incremental interfaces were developed for two intelligent learning environments: ITEM/IP (Brusilovsky, 1993) and ELM-PE (Brusilovsky et al., 1995). Both systems have no model of user knowledge about interface features and use a model of user knowledge on the subject being learned (a programming language) to decide which interface features to enable. For example, if a particular LISP function becomes known to the user of ELM-PE, the button for this function in an adaptive structure editor is enabled.

3 User as Student: Applying Adaptive Sequencing Technology

3.1 The Architecture and the Problem of Sequencing

The work discussed above on adaptive help systems and adaptive interfaces suggests a clear way of implementing both incremental learning and an incremental interface. Two decision-making mechanisms have to be implemented. The "what" mechanism has to decide which unknown interface features the user could and would like to learn next. With incremental learning, these features will be introduced and explained to the user. With the incremental interface, these features will be enabled (and probably explained). The "when" mechanism has to decide whether a particular moment of time is appropriate one at which to interrupt the user and introduce the next set of new features. The work discussed above shows that the "when" mechanism does not pose a real problem. First, it is always possible to implement a simple "when" mechanism on the basis of a small set of domain-dependent heuristics or just to use the non-adaptive DYK style. Second, the mechanism is used only in active help systems. The key problem is the "what" mechanism, or sequencing mechanism. It is used for both passive and active help, and it is where the main "intelligence" of an adaptive interface is.

The work discussed above shows several ways of implementing an adaptive sequencing mechanism. However, we have found that existing experience cannot be applied directly in the Web context because current Web-based interfaces are different. What we call an "interface" in a Web-based application is not a conventional interface. For example, what looks like a button is nothing more than an iconized hyperlink. In essence, a Web-based interface supports hypertext navigation with some amount of form-filling. The problem is that watching what the user is doing in hypermedia provides insufficient information for user modeling (Brusilovsky, 1996). Most known plan-recognition-based techniques of automatic user modeling can hardly be applied in this area. The only information about the user's actions which the system can record is the user's path through the hyperspace and the time spent on each node. The bandwidth of this channel is much lower than in UNIX-like command-based interfaces (Breuker, 1990; Jones et al., 1988; Wolz et al., 1989), action-based interfaces (Encarnação, 1995; Fox et al., 1993) or natural language interfaces (Wilensky et al., 1984).

3.2 Our Solution: The User-as-Student Metaphor

Our solution to the problem of novice users working with complicated interfaces on the Web is to apply the known technologies of incremental interfaces and incremental learning but to use a different sequencing mechanism. Rather then using sequencing based on plan recognition, which is not very suitable for current Web-based interfaces, we use sequencing based on the internal logic of the domain which is represented in a domain model. Domain-model-based sequencing is a classic technology of intelligent tutoring systems (Polson and Richardson, 1988). The heart of this technology is a domain model represented as a network of domain concepts (a concept in ITS is an elementary piece of knowledge). Links between concepts represent the logic of the domain: various semantic relationships between concepts, such as *is-a*, *part-of*, or *prerequisite*. The domain model is used in parallel with an overlay student model of domain knowledge which stores separately the student's level of understanding of each domain model concept. Using domain and student models, an ITS could perform concept sequencing, i.e., build a dynamic personalized sequence of concepts to be learned based on the internal logic of the domain and the student's knowledge. Most advanced ITSs could also perform learning unit sequencing or task sequencing, where a learning unit is an educational task of any kind: an explanation, a test, an example, a problem, etc. (Barr et al., 1976; Brusilovsky, 1992; Capell and Dannenberg, 1993; Vassileva, 1990).

Our idea is to treat a user as a student and an interface as a domain to be learned. Driven by this metaphor, we suggest (1) using an ITS-like domain and student knowledge representation, with each interface feature treated as a domain concept and each hint as a learning unit; (2) applying a well-developed task sequencing technology to build a sequence of interface features to be learned or enabled and a sequence of hints to be presented. We think that this way is a good alternative solution to the sequencing problem for Web-based interfaces in general. Some additional benefits of this solution for InterBook are that this adaptive educational system already has a formalism for domain modeling, student modeling and indexing and that it has some already implemented sequencing mechanisms. In the following sections, we describe briefly the original domain-independent sequencing mechanism implemented in InterBook and explain in detail how we have used this mechanism to implement incremental learning and an incremental interface.

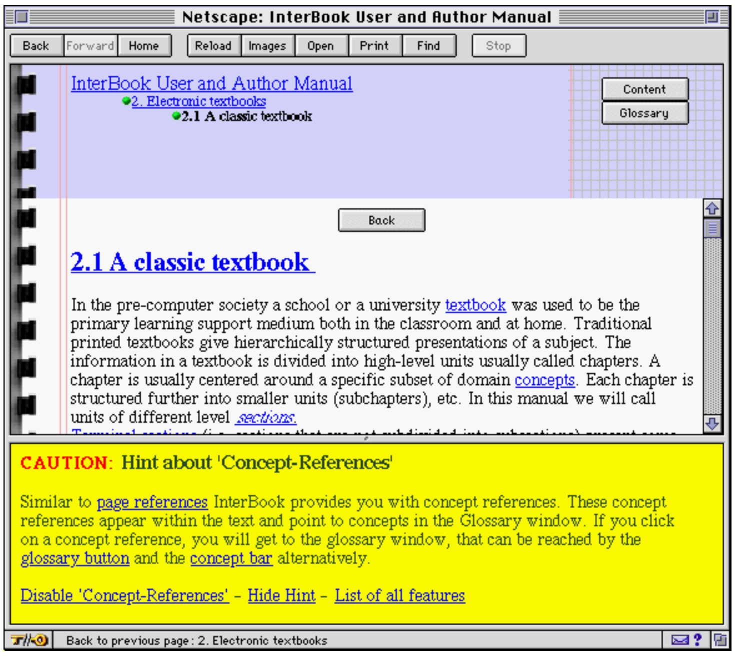
4 An Adaptable and Adaptive Interface in InterBook

As an advanced WWW-application, the InterBook system provides a suitable basis for implementing our ideas. The InterBook system is described in more detail elsewhere (Brusilovsky et al., 1996). Here we provide the minimal description that is required to understand how the incremental interface and the adaptive on-line documentation inherited the adaptation techniques that are already used for adaptive hypermedia in general.

4.1 Adaptive Electronic Textbooks and Techniques

The keys to adaptivity in InterBook are the domain model and the overlay student model. The domain model is a set of concepts that describe the educational domain. All pages of InterBook-based hypertext books can be indexed with domain model concepts by the course author. For each page, an author can provide a list of related concepts, which describe the educational contents in terms of the domain model. Each concept in the index has different roles, being either an outcome concept or a prerequisite of the page. A concept is an outcome concept of a page if this page presents the piece of knowledge designated by the concept. A concept is a prerequisite concept if a user has to know this concept to understand the content of the page. The overlay model keeps track of the individual student's knowledge by assigning to each concept from the domain model scores for reading pages about these concepts and solving tests or problems. It is updated whenever a student reads a related page or solves a test or problem that deals with the particular concept. Indexing is a relatively simple but powerful mechanism, because it provides the system with knowledge about the content of its pages: The system knows which concepts are presented in each section and which concepts have to be learned before starting to learn each section.

Sequencing is one of the applications of the student model in InterBook. The goal of sequencing is to determine the next page that should be learned by the student. This is done in three steps. First, the system computes overall scores for the supposed state of knowledge for each concept. On the basis of these scores, the system can decide whether a concept is already well-learned or should still be the subject of further teaching operations. Second, the system decides for all pages, on the basis of the computed knowledge scores, which pages contain suggested teaching operations or which have missing prerequisites. The results of the second step are used for adaptive annotation of links: Links to concepts and sections of different educational states are annotated by different icons (Figure 1). Third, the system selects the most optimal page from among all available pages that introduce unknown concepts and that are missing no prerequisites. For this purpose, all pages are assigned a certain priority for presentation that is based upon a default value and modified according to the supposed state of knowledge of the required and introduced concepts.



InterBook User and Author Manual
2. Electronic textbooks
2.1 A classic textbook

2.1 A classic textbook

In the pre-computer society a school or a university textbook was used to be the primary learning support medium both in the classroom and at home. Traditional printed textbooks give hierarchically structured presentations of a subject. The information in a textbook is divided into high-level units usually called chapters. A chapter is usually centered around a specific subset of domain concepts. Each chapter is structured further into smaller units (subchapters), etc. In this manual we will call units of different level *sections*.

CAUTION: Hint about 'Concept-References'

Similar to page references InterBook provides you with concept references. These concept references appear within the text and point to concepts in the Glossary window. If you click on a concept reference, you will get to the glossary window, that can be reached by the glossary button and the concept bar alternatively.

Disable 'Concept-References' - Hide Hint - List of all features

Back to previous page: 2. Electronic textbooks

Figure 2. A simplified interface of InterBook that meets the purposes of novices. Disabled features are introduced incrementally with hints. The hint displayed here introduces “Concept References” (Table 1).

4.2 Incremental Interface Learning

According to our original idea of looking at the interface as a domain to be learned, we have created a domain model for the interface. A concept in this model is either an *atomic* interface feature that could be disabled or enabled independently (like “glossary button”) or a more high-level interface-related concept (like “concept-based navigation”). Our interface model has more than fifty interface features. This model is dynamically extendible, since it relies on similar domain-independent mechanisms which are implemented for the electronic textbooks themselves. The interface concepts do not differ from other InterBook concepts that are created by authors of various InterBook-based electronic textbooks. In fact, we have re-indexed the InterBook manual, which is an electronic textbook itself, with a set of interface concepts to integrate the common adaptive electronic textbook with the adaptive interface documentation. In InterBook, incremental learning and interface documentation are implemented with hints (Figure 2). A hint is a textual explanation of one or more interface features (Table 1). Unlike common instructional information, interface documentation has to provide more than “know that” and “know how” aspects of concepts, which explain the location and nature as well as the functionality of the interface features. Interface documentation in an adaptable and adaptive context also has to introduce the user to the “know why” aspect of an interface feature, which explains why the user should switch on and use a certain feature. A hint is presented to the user at the bottom of the corresponding InterBook window, in which the introduced feature is located. Hints appear in separate frames, which are scrollable independently from the main window. This enables the user to work with the original window, which remains unchanged without being disturbed by the hints. Furthermore, this technique binds the hint directly to the relevant window, in which the interface feature is visible. To provide more detailed information about an interface feature, hints can be authored with hyperlinks, which may point to deeper explanations, or related hints, which describe features that might enhance or replace the current feature. In this way, the author of the on-line documentation can create a separate hyperspace of hints for the on-line documentation of the interface.

Table 1. This table illustrates the overall structure of a hint. It implements didactic relations and the documentation itself as well as additional information for event-driven appearance and adaptive sequencing.

Slot	Example	Function
`prerequisites`	Concepts, concept bar	These concepts or interface features must be known before the hint is presented.
`outcome`	Concept hot-links	This feature or concept in introduced in this hint.
`windows`	Textbook window, Glossary window	The hint is appropriate for these windows.
`features`	Concept hot-links, concept bar	These features are relevant to the hint, because the user might want to switch them on or off, when he or she learns about new features and functionalities.
`title`	Hint about ‘‘Glossary window”	The title of the hint by which it can be identified by system and user.
`documentation`	“You have just opened the glossary window...”	The text describing new interface features and their functionality.
`appearance`	“Current page contains a hot-link to a concept”	Rule that describes when a hint should be presented.
`priority`	10	The default priority by which the hint is chosen by the sequencing algorithm.

Hints can be presented in both active and passive modes. Active mode means that hints can appear when a particular window is updated if the sequencing mechanism suggests its presentation. Passive mode means that at any time the user can request a hint himself or herself by using the button “Hint” (Figure 1). This causes the system to present the next most relevant hint. The decision which hint to present is made by the sequencing mechanism. The user can override system's selection by using the “Interface” button, which will bring up a list of all available features. Each feature in this list has attached links, which point to the most relevant hint.

The incremental interface is driven by a simple sequencing mechanism, which chooses a hint if all prerequisites are already known and new outcome concepts are introduced. If the rule that determines the occurrence of a hint applies, the hint will be selected based upon its priority (see Table 1), since several hints might apply in the same situation.

The sequencing mechanism has to made two decisions. The first decision is *what* kind of information the user might be interested in. This decision is made by the general sequencing mechanism based on outcome and prerequisites that is used in InterBook. The second even more important decision, *when* to present this information, is made by simple rules, which enable event-driven occurrence of hints. These rules are implemented by LISP-functions, which are called with an identifier of the contents of the relevant window and the user model. Further, these rules have full access to information about the contents of the window the identifier refers to. This information can be taken into account for the computation of these rules.

An important piece of information for the *when* decision, is the window or interface element a feature might appear in. For instance, features which are specific to the table of contents will be presented only in the contents window. This basic mechanism has been made generic and is explicitly encoded into the structure of a hint (Table 1).

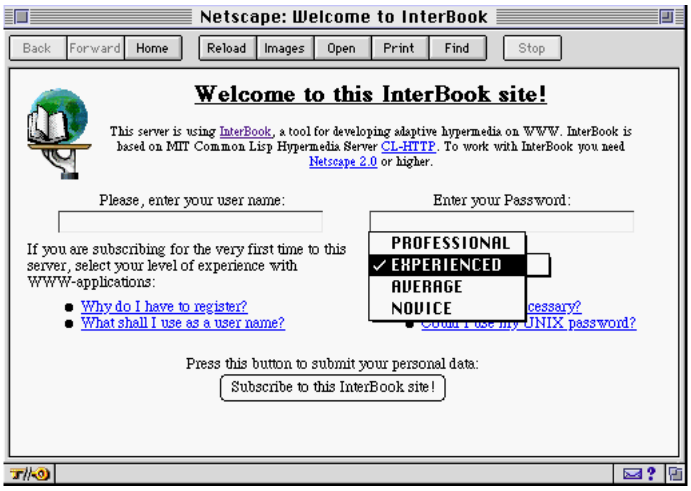
4.3 The Incremental Interface

The incremental interface is linked to incremental learning in that both are integrated into the structure of the hints (see Table 1). Each hint is indexed not only by concepts of the interface domain but also by interface features which are relevant to the hint. For example, when a new feature is introduced, this feature is listed in this slot. Currently, the incremental interface is an option. If the incremental interface is disabled, a hint simply draws the user's attention to new interface features and describes them, leaving the choice to enable them to the user. If the incremental interface is active, the hint also triggers the state of the features that the hint explains. These features are enabled in an InterBook window simultaneously with a display of the hint about them in the bottom frame of this window (Figure 2).

In active or passive mode, users always have the opportunity to enable and disable certain interface features. The features relevant to a hint are listed as links at the bottom of each hint, and their settings can be altered by using those links. Changes in the settings will appear immediately in the assigned window. In this way, the student can systematically explore the look of the interface with and without certain features enabled. Once the user has chosen his settings and deselected the hint, he or she is still able to modify the current settings using the interface button. Selecting this button, the user is provided with a list of all features that are relevant to the current window. Each feature is attached to a link that alters the setting as well as to a link to a relevant hint.

4.4 Default Settings and Stereotyped User Models

In an adaptable and adaptive interface, the user model is divided into two parts: One part tracks users' knowledge about concepts in the overlay model, and another one the current settings for the interface. To meet each user's need from the very beginning, InterBook provides a mechanism for loading both a default overlay model and some default interface settings. The course author has the opportunity to encode both into the subscription page of the system, in which the user has to register him- or herself. During that registration, the user can choose his or her individual level of experience with advanced Web-based applications. Each selection is associated with a stereotype, i.e., a set of domain concepts and default settings, that will be loaded when the user subscribes (Figure 3).



Welcome to this InterBook site!

This server is using InterBook, a tool for developing adaptive hypermedia on WWW. InterBook is based on MIT Common Lisp Hypermedia Server CL-HTTP. To work with InterBook you need Netscape 2.0 or higher.

Please, enter your user name:

Enter your Password:

If you are subscribing for the very first time to this server, select your level of experience with WWW-applications:

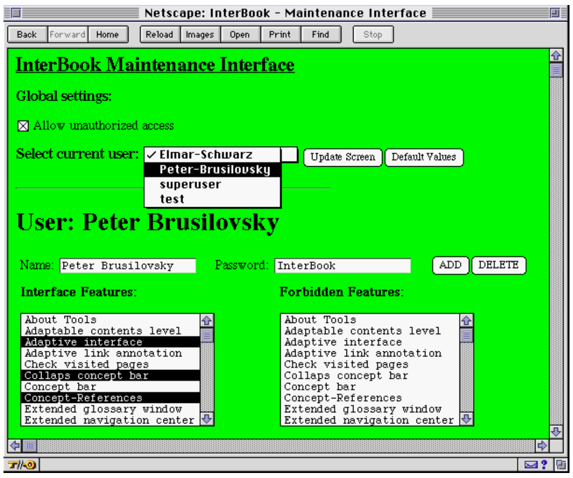
- PROFESSIONAL
- ✓ EXPERIENCED
- AVERAGE
- NOVICE

- Why do I have to register?
- What shall I use as a user name?
- cessary?
- Could I use my UNIX password?

Press this button to submit your personal data:

Subscribe to this InterBook site!

Figure 3. The user must register to allow InterBook to provide individual support. During that dialog, the user can choose his or her individual level of experience with Web-based applications. This selection is associated with default interface settings and the user model.



InterBook Maintenance Interface

Global settings:

☒ Allow unauthorized access

Select current user: ✓ Elmar-Schwarz / Peter-Brusilovsky / superuser / test

User: Peter Brusilovsky

Name: Peter Brusilovsky Password: InterBook

Interface Features:

```
About Tools
Adaptable contents level
Adaptive interface
Adaptive link annotation
Check visited pages
Collaps concept bar
Concept bar
Concept-References
Extended glossary window
Extended navigation center
```

Forbidden Features:

```
About Tools
Adaptable contents level
Adaptive interface
Adaptive link annotation
Check visited pages
Collaps concept bar
Concept bar
Concept-References
Extended glossary window
Extended navigation center
```

Figure 4. Through the maintenance interface of InterBook, instructors can change the current interface settings of individual students.

A second way of changing the user's interface setting from the side of the course author is provided by a maintenance interface. This interface allows the maintainer of the server to alter each student's setting at any moment via the WWW. Each interface setting can be either enabled or disabled—or even marked as being forbidden for certain students, if the teacher dislikes certain interface features of InterBook (Figure 4).

5 Conclusion

This paper is centered around the problem of novice users working with complex interfaces of advanced Web-based applications. It investigates two possible adaptive ways to help these users: incremental learning and an incremental interface. An analysis of existing relevant research shows that the key problem to be solved is how to build an adaptive sequence of interface features to introduce or to enable. Classic plan-recognition-based sequencing mechanisms are not very suitable in the WWW context. The main theoretical contribution of the paper is the suggestion to use a domain-logic-driven sequencing mechanism which is known in the area of intelligent tutoring systems. The slogan is to treat a user as a student and an interface as a domain to be learned. It implies that we use an ITS-like knowledge representation with an interface feature treated as a domain concept and a hint as a learning unit and to apply task sequencing technology developed in the area of ITS. The practical contribution of the paper is an implementation of domain-logic-driven sequencing of hints in InterBook, a shell for developing adaptive electronic textbooks on the WWW. InterBook is an intelligent educational system and it has an embedded sequencing mechanism which was originally applied for course sequencing. In the second part of the paper we described briefly the original domain-independent sequencing mechanism implemented in InterBook and explained in detail how we have re-used this mechanism to implement incremental learning and an incremental interface. We think that InterBook provides a good model for other researchers working on adaptive interfaces for Web-based applications.

References

Barr, A., Beard, M., and Atkinson, R. C. (1976). The computer as tutorial laboratory: The Stanford BIP project. *International Journal of Man-Machine Studies* 8:567-596.

Breuker, J. (1988). Coaching in help systems. In Self, J., ed., *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction*. London: Chapman and Hall. 310-337.

Breuker, J. (1990). *EUROHELP: Developing Intelligent Help Systems*. Final Report on the P280 ESPRIT Project EUROHELP. Copenhagen: EC.

Brusilovsky, P. (1993). Student as user: Towards an adaptive interface for an intelligent learning environment. In Brna, P., Ohlsson, S., and Pain, H., eds., *Proceedings of AI-ED'93, World Conference on Artificial Intelligence in Education*. AACE. 386-393.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction* 6:87-129.

Brusilovsky, P., Schwarz, E., and Weber, G. (1996). A tool for developing adaptive electronic textbooks on WWW. In Maurer, H., ed., *Proceedings of WebNet'96, World Conference of the Web Society*. AACE. 64-69.

Brusilovsky, P., Specht, M., and Weber, G. (1995). Towards adaptive learning environments. In Huber-Wäschle, F., Schauer, H., and Widmayer P., eds., *Herausforderungen eines globalen Informationsverbundes für die Informatik: GISI 95*. Berlin: Springer. 322-329.

Brusilovsky, P. L. (1992). A framework for intelligent knowledge sequencing and task sequencing. In Frasson C., Gauthier G., and McCalla, G. I. (eds.), *Intelligent Tutoring Systems: Proceedings of Second International Conference, ITS'92*. Berlin: Springer-Verlag. 499-506.

Burton, R. R., and Brown, J. S. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies* 11:5-24.

Capell, P. and Dannenberg, R. B. (1993). Instructional design and intelligent tutoring: Theory and the precision of design. *Journal of Artificial Intelligence in Education* 4:95-121.

Chin, D. N. (1989). KNOME: Modeling what the user knows in UC. In Kobsa, A., and Wahlster, W., eds., *User Models in Dialog Systems*. Berlin: Springer. 74-107

Dieterich, H., Malinowski, U., Kühme, T., and Schneider-Hufschmidt, M. (1993). State of the art in adaptive user interfaces. In Schneider-Hufschmidt, M., Kühme, T., and Malinowski, U., eds., *Adaptive User Interfaces: Principles and Practice*. Amsterdam: North-Holland. 13-48

Encarnação, L. M. (1995). Adaptivity in graphical user interfaces: An experimental framework. *Computers and Graphics* 19:873-884.

Fischer, G. (1988). Enhancing incremental learning process with knowledge-based systems. In Mandl, H., and Lesgold, A., eds., *Learning Issues for Intelligent Tutoring Systems*. New York: Springer. 138-163.

Fox, T., Grunst, G., and Quast, K.-J. (1993). *HyPlan–A context-sensitive hypermedia help system*. Available as Report 743, GMD, St. Augustin, Germany.

Goodman, B. A., and Litman, D. J. (1992). On the interaction between plan recognition and intelligent interfaces. *User Modeling and User-Adapted Interaction* 2:83-115.

Hoppe, H. U. (1993). Intelligent user support based on task models. In Schneider-Hufschmidt, M., Kühme, T., and Malinowski, U., eds., *Adaptive User Interfaces: Principles and Practice*. Amsterdam: North-Holland. 167-181.

Jones, J., Millington, M., and Ross, P. (1988). Understanding user behavior in command-driven systems. In Self, J., ed., *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction*. London: Chapman and Hall. 226-235.

Matthews, M. M., and Nolan, T. (1985). Levi: A prototype active assistance interface. *Proceedings of USENIX Association Summer Conference,* 437-454.

Nessen, E. (1989). SC-UM: User modelling in the SINIX consultant. *Applied Artificial Intelligence* 3:33-44.

Polson, M. C., and Richardson, J. J., eds. (1988). *Foundations of Intelligent Tutoring Systems*. Hillsdale, NJ: Erlbaum.

van der Veer, G. C. (1990). *Human-Computer Interaction. Learning, Individual Differences and Design Recomendations*. Amsterdam: Vrije Universiteit.

Vassileva, J. (1990). An architecture and methodology for creating a domain-independent, plan-based intelligent tutoring system. *Educational and Training Technology International* 27:386-397.

Wasson, B. and Akselsen, S. (1992). An overview of on-line assistance: From on-line documentation to intelligent help and training. The *Knowledge Engeneering Review* 7.

Wilensky, R., Arens, Y., and Chin, D. (1984). Talking to UNIX in English: An overview of an on-line UNIX consultant. *Communications of the ACM* 27:574-593.

Winkels, R. G. F. (1990). *User Modelling in Help Systems*. Berlin: Springer. 184-193.

Wolz, U., McKeown, K. R., and Kaiser, G. E. (1989). Automated tutoring in interactive environments: A task-centered approach. *Machine Mediated Learning* 3:53-79.

Zissos, A. J. and Witten, I. H. (1985). User modelling for a computer coach: A case study. *International Journal of Man-Machine Studies* 23:729-750.