Adaptive Hypermedia System for Supporting Information Providers in Directing Users through Hyperspace

Yoshinori Hijikata, Tetsuya Yoshida and Shogo Nishida

Graduate School of Engineering Science, Osaka University, JP
E-mail:hijikata@nishilab.sys.es.osaka-u.ac.jp

ABSTRACT

This paper introduces an adaptive hypermedia system that provides a means for information providers to direct their users through hyperspace. With our system, the information providers are able to easily direct users to their own personalized navigation paths. Destination options are determined by hiding links and by applying rules so that the users are offered the best paths for them. In order to reduce the information providers' efforts in creating navigation rules, we simplified the format of these rules and offer an authoring tool that verifies the navigation rules and reports any errors if they exist. This tool not only allows the information providers to easily write navigation rules but also guarantees the adequacy of the navigation path.

KEYWORDS: adaptive hypremedia, navigation, rule, link hiding, authoring tool

INTRODUCTION

The WWW (World-Wide Web) has become popular as a tool for information retrieval. Furthermore its applications are diversifying into such areas as electronic commerce, marketing and education [14]. In these kinds of application, the information providers or the Web masters often want to direct their users through hyperspace as desired according to each user's preferences and status. For example, to offer teaching materials according to the learner's knowledge or study history, to regulate obscene contents to prevent children from viewing them, and to guide marketing campaign so that banner ads and item recommendations are in accord with each customer's preferences.

Some researchers have been studying these kinds of user navigation aids in the research field of adaptive hypermedia [1]. Adaptive hypermedia is hypermedia with functions to dynamically adapt to each user. A hypermedia document is the basic component of the WWW and allows users to freely move and retrieve information in hyperspace, which consists of nodes containing information and links relating the nodes. Adaptive hypermedia systems realize their user-adaptations based on various kinds of user information such as the user's prior knowledge, objectives and interests. Many adaptive hypermedia systems implement user-navigation guidance created by the information providers, using methods that allow the information providers to describe the navigation rules according to user categories and user behaviors defined in advance[2, 4, 5, 6, 7, 8, 9, 10, 13]. However this method leads to the following problems:

1. Information providers have great difficulty describing the navigation rules as they move towards fine-grained navigation control.
2. It becomes difficult to predict the resulting states for various kinds of users, because the navigation dynamically varies according to each user.

These problems become more critical as the need to direct users accurately increases.

This research aims to construct an adaptive hypermedia system that reduces the burden on information providers and prevents errors in the described navigation rules. We propose a system solving the above problems by the following mechanisms:

1. A simplified format for navigation rules.
2. An authoring tool that examines the navigation rules.

Existing hypermedia systems do not focus on providing mechanisms and functions that can reduce the burden on information providers for creating and verifying navigation rules.

In the proposed system, (1) destination options are determined by hiding links so that information providers can direct users accurately, (2) long-term and short-term user information can be used in the navigation rule, because it is generally important to consider users from both perspectives [12], and (3) the format of user information is also simple, because the format of navigation rule is simple.

This paper is organized as follows. First, we introduce existing navigation methods and user models, and explain the reason why we adopted link hiding and the type of user information used in our system. After that we describe the navigation method and rule format of our system and explain the authoring tool. Next we describe the implementation of the system and its evaluation. Finally we offer some conclusions.

ADAPTATION METHOD AND USER MODEL

Existing adaptive hypermedia systems construct a user model and use it for adapting to each user [1]. The user models describe information about the users such as the users' knowledge, objectives, and interests. This section describes the adaptation method and the user model.

Adaptation method

Adaptation methods can be classified into content-level adaptation and link-level adaptation[1]. Content-level adaptation adapts the displayed content of the node. Link-level adaptation adapts the links of the node. We adopted link-level adaptation, because the focus of this research is on how information providers can direct users through hyperspace.

Link-level adaptation can be classified into the four types (direct guidance, adaptive ordering, hiding, adaptive annotation) according to how the links are modified [1]. Direct guidance attaches an explanation to the link the user should follow or inserts a [Next Link] button for directing the user. Adaptive ordering sorts links in the order of the degree of suitability to the user. Hiding narrows the accessible hyperspace by hiding links. Adaptive annotation attaches additional decorations such as icons and colors to the links.

Out of these four kinds of link-level adaptation, direct guidance, adaptive ordering and adaptive annotation may display links that the information provider does not recommend. Although this leads to the problem the user may not always follow the information provider's intentioned navigation paths, it also gives the user the freedom to select links the information provider does not recommend. These approaches are especially suitable for applications like (1) information retrieval systems and (2) learning systems that focus on the learners' active information retrieval.

The link hiding method does not display any extraneous links. Although this forces the user to follow the information provider's navigation paths, it has the corresponding advantage that the information provider can always direct the users as desired. This constrained approach is suitable for applications such as (1) learning systems for business training where the learner follows the information provider's directions to acquire the knowledge quickly, (2) help systems for application software, and (3) information systems that screen obscene content from children. We adopted link hiding because our system focuses on the situation where the information provider wants to direct users precisely.

User model

As user models for adaptive hypermedia, there are overlay models, stereotype models, and models using keywords[1]. An overlay model is based on a structural domain model which is represented as a semantic network of domain concepts. A stereotype model assigns the user to one of several possible stereotypes for each dimension of classification. A model using keywords is represented as a vector or a matrix whose elements are the degree of interest in a keyword or topic. Although it is simple as a model, a browsing history expressed with the permutation of URL is also used for some Web applications. This is useful for describing the user's recent browsing action.

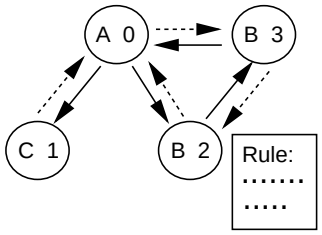

Figure 1: Hypermedia model.

Since it is generally important to consider long-term and short-term user information for designing user models [12], we also use both kinds of user information. Since our system simplifies the format of navigation rule, we also make our user model simple. We use pairs consisting of a property and a value (we call "user parameter") for modeling the user's long-term information. This approach is actually similar to all three kinds of user model described above. We also use a browsing history represented as the sequence of the node classifications (we call "path history") for modeling the user's short-term information.

NAVIGATION METHOD

Hypermedia model

Figure 1 shows the hypermedia model used in our system. The circles in the figure show hypermedia nodes. The content displayed for the users are contained in these nodes. The user can move between nodes by following a link represented as a straight line with an arrowhead. We also define a return link represented as a dotted line with an arrowhead. Although the user cannot follow this link, this is required for implementing the authoring tool (explained in the next section). The identity of the node is represented by a number and node classes (explained later) are represented by alphabetic characters. Some of the nodes have navigation rules (explained later) as created by the information provider.

Class

Every node has a "class" represented by symbols such as alphabetic characters. The reason why we introduced the notion of class is to allow information providers to describe the navigation rules by generalizing and specializing the characteristics and meanings of the navigation rules. Class is used for representing the user model and the navigation rules. The information provider defines a class in terms of the following distinctions:

1. Whether or not the system displays the contents for a specific kind of user.

2. Whether or not the node offers users an explanation, asks a question, or does something else for an educational purpose.

3. Which of several categories of contents the node belongs to (in cases where the information provider deals with information in more than one category).

Representation of user model

The user parameter is represented as a value from some range. The information provider assigns a meaning to the user parameter according to his/her navigation control. The user, the information provider, or some other person sets the value of a parameter. For example, they can be set by asking users to answer a questionnaire or by using the results of regular paper tests in academic environments.

The path history is represented concisely as the sequence of classes of the nodes the user has visited, and indicates the order of information the user has browsed. If the user browsed nodes in an order such that their classes were $C \rightarrow A \rightarrow B \rightarrow B \rightarrow A$, the path history is represented as $CABBA$.

Navigation rule

This system decides which links to hide based on a navigation rule that may be associated with the current node. There are four kinds of navigation rules: (1) node path rules, (2) general path rules, (3) node user rules, (4) general user rules.

A navigation rule that uses a path history is called a path rule and a navigation rule that uses a user parameter is called a user rule. The navigation rule can also be classified into two types, node rules and general rules. A node rule is defined and applied only for a specific node. A general rule is for describing frequently followed navigation paths in hyperspace and frequently used segmentation of the range of the user parameter. The information provider can apply it for any node.

In a navigation rule, the information provider should describe the links that should be displayed by the node ID or by the class of the node that is the target of the link. The system hides all links that are not referenced in a navigation rule as links to be displayed. The format of these four kinds of navigation rule is as follows:

1. Node path rule

$$C_{11} \cdots C_{1h} + \cdots + C_{m1} \cdots C_{mh} = D_1, \cdots, D_n \quad (1)$$

2. General path rule

$$C_{11} \cdots C_{1h} + \cdots + C_{m1} \cdots C_{mh} = C_1^f, \cdots, C_n^f \quad (2)$$

3. Node user rule

$$e_1 \# P_i \# e_2 : D_1, D_2, \cdots, D_n \quad (3)$$

4. General user rule

$$e_1 \# P_i \# e_2 : C_1^f, C_2^f, \cdots, C_n^f \quad (4)$$

The above variables represent the following:
C: Class
D: Id of the node to be shown
C^f: Class of the node to be shown
h: Number of histories that will be referred to
m: Number of path patterns
n: Number of IDs or classes of nodes to be shown
P_i: The ith user parameter (property)
e: Boundary number of the user parameter
The symbol '$\#$' represents one of the following three operators: $<$, $\leq$ or $=$.

$C_{m1} \cdots C_{mh}$ in the path rules (1) and (2) shows the path history pattern, which represents the order of the user's search in hyperspace as a permutation. The path rule means that the system displays links whose node ID or class is described on the right part of the rule if the user's path history matches one of the path history patterns which are described on the left side. In user rules (3) and (4), $e_1 \# P_i \# e_2$ specifies the user parameter P_i and the applicable range of the parameter values. The user rule means that the system displays links whose node ID or class is described on the right side if the user parameter specified on the left side is within the specified range.

If a node has several navigation rules, the system displays all links that any navigation rule accepts. This means that if at least one rule out of several rules approves the display of a specific link, the system displays the link regardless of the other navigation rules.

Example of navigation

Figure 2 shows an example of navigation using path rules and user rules. For an educational application, the classes are defined as follows:

- A: Nodes with a question.
- B: Nodes which display an appropriate response when the user answers correctly.
- C: Nodes which display an appropriate response when the user answers incorrectly.
- D: Nodes which display an explanation for students with good school records.
- E: Nodes which display an explanation for students with poor school records.

We assume that the user parameter stands for the knowledge level on a specific subject and is set based on the result of a standard paper examination at the school.

A node path rule is defined for Node No. 5. This rule is only applicable at this node. "$ACA = 7$" in this rule means that when the user comes to Node No. 5 and the user's path history is ACA, the system shows the link to Node No. 7 and

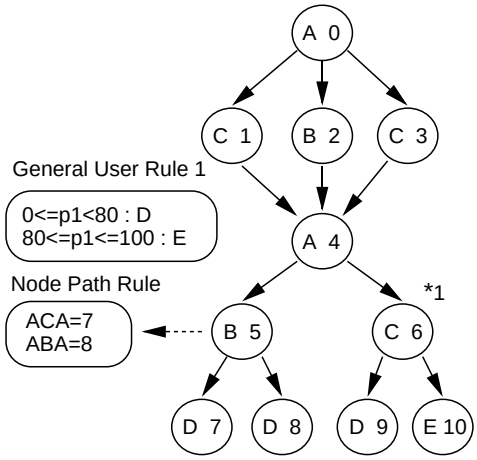

Figure 2: An example of the rules.

hides the link to Node No. 8. Because Class B means that the user answered correctly and Class C means that the user answered incorrectly, the history in this order means that the user answered the question in Node No. 0 incorrectly and answered the question in Node No. 4 correctly. "$ABA = 8$" means that if the user answered correctly for the question in Node No. 0 and also answered the question in Node No.4 correctly, then the system shows only the link to Node No. 8. That is to say the system changes the teaching materials according to the results of the previous questions.

A general user rule is also defined. This rule can be applied at any node in the hyperspace and in this case Node No. 6 uses it. In this general user rule, if the User Parameter 1 is 0, or more than 0 and not exceeding 80, the system shows any links to nodes whose class is D and ignores other links. If the user parameter is more than 79 and not exceeding 100, the system shows the link to nodes whose class is E and ignores other links. Because User Parameter 1 refers to the user's knowledge level of a specific subject, this means the system can offer suitable teaching materials based on the student's ability.

AUTHORING TOOL

Objective

Generally an authoring tool is important for an adaptive hypermedia system so that the information provider can direct users in hyperspace [3, 5, 6, 11]. Therefore our system provides an authoring tool that helps the information provider in describing the navigation rules. This tool examines the execution results of the navigation rule before they are incorporated into nodes. This aims for correct navigation with fewer errors and for simplification of the information provider's efforts to describe the navigation rules. We focused on detecting the following two kinds of problems because they can happen in any kind of content and are very likely to be related to navigation errors:

1. Dead end: There is a possibility that all links are hidden and the user cannot go anywhere after reaching a node with a navigation rule. This dead end problem could be caused by a bad navigation rule. Were a dead end to appear, it would force the user to stop searching in hyperspace. This may create obstacles to the user's progress.
2. Loop: In some navigation, the user may reach a node where the user has already been. We call this search looping. As seen in the WWW, we can use a loop effectively, for example as a link for returning to a top page, and it has an important role. However our concerns are that there may be unintended loops or the user may not be able to follow a loop that the information provider intended the users to follow. This is because the system hides links dynamically, which could cause problems for user navigation.

Dead end detection

A dead end can be caused by a path rule or a user rule or by a set of rules. This section describes an algorithm that checks if a dead end will happen in a node (or if there is a possibility a dead end can happen in the node) because of the path rules or user rules. In our system, if a node has several kinds of navigation rules, the system displays all links that any rule tries to display. It is possible that even if the tool detects a dead end caused by one kind of navigation rule (e.g. a path rule) in a node, the other kind of rule (e.g. a user rule) may try to display links in the node. Therefore when the tool detects an apparent dead end at a node, it checks whether another kind of rule is defined. If no other rule is defined for the node, it has detected a dead end. If another kind of rule is defined on the node, it has detected the possibility of a dead end.

Detection of dead ends caused by path rules A dead end caused by path rules happens when (1) the path history pattern the user has followed is not included in the path rules or (2) none of the links of the current node are described in the path rules as displayable, based on the path history pattern the user has followed to reach the current node. Here is an algorithm to check whether a dead end caused by path rules will happen or whether there is a possibility that it will happen in a specific node. This algorithm not only detects dead ends (dead ends possibility) but outputs the path history pattern that causes a dead end.

Detection algorithm for dead ends caused by path rules:

1. Node specification: The information provider specifies the node he/she wants to check.
2. Examination of displayable links: The system checks whether or not the links to be displayed according to the path history pattern described in the path rule defined at the specified node really exist in hyperspace. This is checked by comparing the nodes described as displayable in the path rules with the nodes that are the targets of the links of the current node.
3. Registration of live path: The system recognizes the path history pattern, which has links which should be displayed and

really exist, as a live path (If the user follows the live path to the specified node, there are links to proceed). It registers the live path in a list according to the length of the path history pattern. We call this list the live path list.

4. Depth-first search: The system executes a depth-first search from the current node (It is the specified node at first) using the return links mentioned in the last section and considering the current node to be the root of the inverted tree.
5. Path examination: The system refers to the live path list based on the length of the current depth-first search and checks whether or not the current path of the depth-first search is a live path for that node. If it is a live path, the system does not search deeper on this path, but returns to Step 4 for continuing the depth-first search from the upper node. If it is not a live path, the system continues to Step 6.
6. Detection of dead end possibility: If the length of the current depth-first search is the maximum length of the paths registered in the live path list, the system has determined that there is a possibility that a dead end happens when the user follows this path and the search continues to Step 7. If it is not the maximum length, the system returns to Step 4.
7. Decision on dead end: The system checks if a rule is defined at any of the nodes on the path. If no navigation rule is defined for any of these nodes, the system has determined that a dead end happens when the user has followed this path. If navigation rules are defined for at least one node, the system has determined that there is a possibility that a dead end happens when the user has followed this path. After that the system returns to Step 4.

Figure 3 shows an execution example of this algorithm. This example tries to detect a dead end at the shaded node in the figure. In this case, only the shaded node has navigation rules and the other nodes do not have any navigation rules. Out of 6 path history patterns in the navigation rule, only AB, AA, ABA, CCA have links which can be displayed and really exist. The system registers these path history patterns as live paths. After that, the system executes the depth-first search and dead end detection. In this example, the path CCB is not a live path. The system determines a dead end happens if the user follows this path, because the length of this path is the maximum length of the live paths in the live path list and there are no nodes that have a navigation rule in the path. CCA is an example of a path that does not cause a dead end, because it is a live path.

Detection of dead end caused by user rules A dead end caused by user rules happens when (1) the values of the user's user parameters are not within the range described in the user rules or (2) all displayable nodes described in the user rules do not exist as target nodes of links of the node where the user is. Here is an algorithm to check whether a dead end caused by user rules will happen in a specific node. This algorithm not only detects dead ends but also outputs the rule that causes a dead end.

Detection algorithm for dead ends caused by user rules:

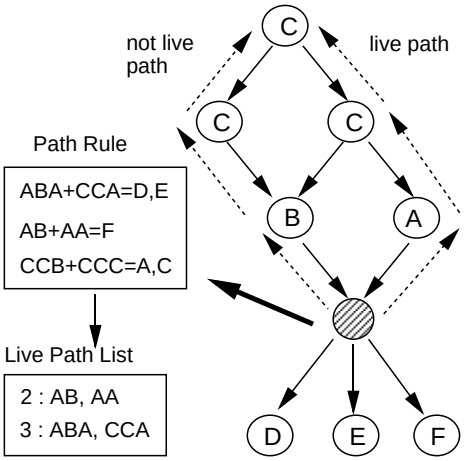

Figure 3: An example of dead end detection.

1. Node specification: The information provider specifies the node he/she wants to check.
2. Examination of displayable link: The system checks whether or not the displayable link for a specific range of the user parameter as described in the user rule of the specified node really exists. This is checked by comparing the displayable nodes in the user rules with the nodes that are the targets of the links of the current node. If such links exist, it recognizes the range as a live range (a range that has displayable links).
3. Examination of the range of the parameter: The system checks whether or not all ranges of the user parameter are live ranges. If there is a range that is not a live range, it has determined that a dead end will occur within that range.

Loops

Even if a path defines a loop without considering the effects of link hiding, it may not be a loop after link hiding is taken into account. It is necessary to set a specific user parameter and follow the path according to the navigation rule to check if the loop becomes a loop after link hiding. Here is an algorithm to detect loops by doing depth-first search from a specific node. This algorithm not only detects dead ends but also outputs the path of the loop.

Loop detection algorithm:

1. Node and maximum length specification: The information provider specifies the node he/she wants to start the depth-first search from and the maximum length of depth-first search.
2. Navigation rule execution: The system executes the navigation rules of the current node and hides links. After that it registers the displayed links in a displayed link list, which is necessary to execute depth-first search only using the displayed links.
3. Depth-first search execution: Perform one step in a depth-first search using a link registered in the displayed link list.

4. Loop examination: The system searches for the node ID of the node it has reached now in the path history of the depth-first search. If the same node ID exists in the path history, the system has determined the path from the previous node which has the same node ID to the current node is a loop.
5. Length check: If the search length of depth-first search is the length specified in Step 1, the system goes back to the previous node and returns to Step 3. Otherwise it returns to Step 2.

In Step 2 of the above algorithm, if a path rule is defined on the node where the system has reached and the search length of depth-first search is shorter than the length of the path history pattern described in the rule, the system cannot execute a path rule. In this case, the system does not execute the path rule and displays all links for detecting all possibilities for loops.

IMPLEMENTATION AND EVALUATION

Implementation of the system

We implemented the system using the C++ language. In the system, the information provider can use 16 kinds of classes. The maximum length of the path history is 16. There are no limits on the other parameters, the number of node, the number of links, the number of rules, and so on.

Figure 4-(a) shows an example of the system when the user searches the hyperspace. The user browses text information and searches by inputting the number of the link. Figures 4-(b,c) shows examples of the authoring tool. Figure 4-(b) is an example of dead end detection. It shows the sequences of the node IDs and the classes of the path history patterns leading to the information provider's specified node and causing dead end. Figure 4-(c) is an example of loop detection. It shows the sequences of the node IDs of the detected loop.

```
(1)
Learning System for Computer  Science
1 System Development
2 Operating System
>1
(26)
Question 1  What is the name of system development
method using the following steps.
Analysis of Requirement -> Requirement Definition ->
System Design -> Program Design -> Programming -> Test
-> Employment -> Maintenance
1 Prototype model
2 User model
3 Water fall model
>
```

(a) Example of search by the user

```
Dead end detection mode
Input target node for detection>15
Input a rule type for detection
1 Path rule   2 User rule
>1
Dead end will occur in the following path.
3->6->9->10->11
C->A->B->A->C
```

(b) Example of dead end detection

```
Loop detection mode
Input start node for detection>83
Maximum length of loop>5
Following path will be a loop.
1->24->48->1
```

(c) Example of loop detetion

Figure 4: Output example from the system.

Objective of evaluation

We evaluated the system from the following viewpoints:

1. Qualitative evaluation of the entire system: This evaluation looks at how the features of the system, which are the simple rule format, the authoring tool, and adaptation by link hiding, appeared to the information providers. We asked some information providers to use this system and give us their subjective opinion on the effectiveness of the entire system.
2. Quantitative evaluation of the authoring tool: This evaluation examines whether the authoring tool succeeds in reducing the information providers' efforts to describe the navigation rules and insuring correct navigation. We quantitatively evaluated whether the authoring tool reduced the time that the information provider required for describing the navigation rules (the description time) and reduced the number of errors in the described navigation rules.

Evaluation method

Qualitative evaluation of the entire system Five information providers created content and described navigation rules for the content. After that they gave us their subjective opinions on the system's effectiveness and problems. They created the following content:

- Information Provider A: Educational content for science.
- Information Provider B: Educational content for English.
- Information Provider C: Content included obscene parts that children were not to see.
- Information Provider D: Content for marketing.
- Information Provider E: Content for software on-line manual.

Quantitative evaluation of the authoring tool Ten information providers participated in the experiment as subjects. These subjects were divided into two groups. The subjects of one group (Group A) described navigation rules without the authoring tool. The subjects of the other group (Group B) described navigation rules with the authoring tool. We evaluated the authoring tool based on the description time and the error ratio in the described navigation rules. The procedure of the experiment was as follows:

1. Experiment preparation: The experimenter prepared the experiment in the following way:

(a) Prepare content as hypermedia data.

(b) Assign meanings to the user parameters.

(c) Assign meanings to the classes.

(d) Define the class of every node.

(e) Create the task for the experiment (the navigation rules the subjects should create).

2. Explanation for the subjects: The experimenter explained how to describe the navigation rule to both groups, and how to use the authoring tool to Group B. The experimenter asked the subjects to work on a practice task for getting used to the system. After that the experimenter explained the task for the experiment. The experimenter sat by the subject and answered the subject's questions, but did not provide direct hints or solutions for the experimental task.
3. Experiment: Each test subject worked on the task and described all navigation rules. The experimenter observed the subjects working on the task during the experiment and measured the times taken for the descriptions.
4. Analysis: The experimenter measured the results of the experiment in the following way:

(a) Execute the navigation rule described by the subject and check (1) whether or not there is an error, (2) whether or not there is a dead end, and (3) whether or not there is an error in the loop when the navigation includes a loop.

(b) Calculate the following three evaluation parameters: (1) error ratio, which is the ratio of the tasks with an error in relation to all tasks, (2) dead end ratio, which is the ratio of the tasks with a dead end in relation to all tasks, and (3) loop error ratio, which is the ratio of the tasks with a loop and an error in relation to all tasks with loop.

(c) Determine the relative effectiveness of the authoring tool as it affects the above three evaluation parameters.

The content we created for the experiment are intended for students who study computer science. The size of the content, the usage of the user parameters and classes, and the contents of the navigation task are shown in the appendix.

Evaluation result

Qualitative evaluation of the entire system The information providers offered the following subjective opinions about the system:

1. I did not need programming knowledge and could describe the navigation rules easily because the rule format is simple.
2. Users will not hesitate to select links because I hid all the unnecessary links.
3. When I created the navigation with a loop, I had to check if the user can follow all of the paths in the loop. However the authoring tool showed all the paths of the loop and I did not have to follow all of the paths by myself.
4. The navigation rule with a dead end that was discovered by the authoring tool also had other errors.

Opinion 1 shows that even information providers who do not have the programming knowledge accepted the navigation rule description, because the rule itself is simple. Opinion 2 shows that link hiding reduced the users' hesitations in hyperspace and gave the information provider confidence that he could correctly direct the users. Opinion 3 shows that the information providers could recognize whether the loop paths they created were or were not accessible at a glance, because the authoring tool displays the sequences of the node IDs of all of the loop paths. From Opinion 4, we think that the navigation rule itself is more complex or the information provider created rules more carelessly in the node that has a dead end than in the other nodes.

The subjects also pointed out the following problems:

1. I have to describe the path history in a path rule even if I just want to create an easy navigation rule that only checks whether or not the user has passed a specific node.
2. I cannot change the user parameters while the user is searching in the hyperspace.
3. I think if the system had some general rules for frequent usage prepared in advance, I could describe the navigation rules faster.
4. I have to check not only dead ends and loops but also the detailed result of the navigation to see the sets of displayed links and sets of hidden links according to a specific path. Without this I have to do simulation by setting user parameters and following the paths.

Solving Problem 1 and Problem 2 has the advantage of strengthening the descriptive capability of the navigation rules. One of the solutions for Problem 1 is providing special user parameters for temporary flags and rules for updating the special user parameters. However this requires the information provider to manage the flags. The system should support the management of the flags. As regards Problem 2, we believe the system should not easily update the long-term user information (user parameters), because of the need to maintain the users' trust of our user model. However if the contents of the hypermedia are refined enough to manage changing the user parameters, there should be little problem when the navigation rules change them. In this case, the information provider should have the responsibility for the appropriateness of the content and the rules for updating the user parameters, because the system cannot guarantee the appropriateness of them.

The general rules for frequent usage mentioned in Problem 3 are important because the information provider can not only use them but also refer to them. We will provide them as a navigation rule library for our system. Providing other navigation rule checking functions besides dead end detection and loop detection would be a solution for Problem 4. However we only provided dead end and loop detection in the current version of the authoring tool for our system. The reason is that the information provider can perform simulations

by himself or herself, yet it is hard to manually detect dead ends and loops. We are considering functions besides dead end detection and loop detection to enhance the authoring tool.

Quantitative evaluation of the authoring tool As shown in Tables 1,2,3,4, Subjects a-e were in Group A and Subjects f-j were in Group B. Table 1 shows the description time. We did an analysis of variance to determine whether there is a significant difference in the description time between the two groups. However there was not a significant difference at the 5% level of significance.

Table 2 shows whether or not the subject described the navigation rules without an error, and the error ratio. Table 3 shows whether or not the navigation rule that the subject described included dead ends, and the dead end ratio. Table 4 shows whether or not the navigation rules (only for Tasks 5 and 6) that the subjects described included loop errors, and the loop error ratio. In each table, a circle shows that there is no error, there is no dead end, or there is no loop error. An X shows that there are errors, dead ends, or loop errors. Although the value of the error ratio, dead end ratio, and loop error ratio assumes discrete values because the number of tasks is small, we did an analysis of variance on these three parameters to get an idea of the effectiveness of the system. The result is that there is a significant difference between the two groups at the 5% level of significance in the above three parameters.

Figure 5 is a graph of the relationship between the description time and the error ratio. As regards the description time and the error ratio, the correlation coefficient of the group A is -0.67 and the correlation coefficient of the group B is -0.89. Although we cannot guarantee high and negative correlation, we see an apparent relationship that as the description time becomes longer the error ratio gets smaller.

The overall results, showing significant differences in the error ratio, dead end ratio, and loop error ratio, indicate that the authoring tool reduced the numbers of navigation errors. There is not a significant difference in the description time. We think the reason is that the Group B subjects tended to rely on the authoring tool to check the described navigation rule. However if the subject uses the authoring tool, the dead end and loop detection shows whether or not there is an error, and they repeatedly modified the navigation rules and checked them. The reason that there is not a significant difference in the overall description time is that (1) there are individual differences in the description times, (2) the authoring tool reduced the time to check the described navigation rule, and (3) the Group B subjects spent time repeatedly modifying and checking the navigation rules, thereby offsetting the time saved during each check. However we can recognize the effectiveness of the authoring tool also on the description time, because of the fact that the description time tends to get longer as the error ratio becomes smaller and the error ratio becomes smaller when the subjects use the authoring tool. This again shows that the authoring tool reduced the information providers' efforts in describing the navigation rules.

Table 1: Description Time.

Subject	Time(min)	Subject	Time(min)
a	59.8	f	58.4
b	43.2	g	69.2
c	61.3	h	45.7
d	52.5	i	50.0
e	52.5	j	71.4

Table 2: Error in the navigation rule.

	Task						Error
Subject	1	2	3	4	5	6	ratio(%)
a	O	O	O	O	X	O	17
b	O	O	O	O	X	X	33
c	O	X	O	O	X	O	33
d	O	X	O	O	X	O	33
e	O	O	O	O	X	O	17
f	O	O	O	O	O	O	0
g	O	O	O	O	O	O	0
h	O	O	O	O	X	O	17
i	O	X	O	O	O	O	17
j	O	O	O	O	O	O	0

Discussion

We implemented an adaptive hypermedia system for information providers to properly guide users in hyperspace. Our system uses link hiding as the primary adaptation method and prevents users from selecting links that information providers do not make available. Our evaluation shows that adding a supporting tool that checks the execution of link hiding for this function enables information providers to direct users more reliably.

Because of the problems information providers have in describing the navigation rules, we focused on the following:

1. The effort required to express the navigation according to the format of the navigation rules.
2. The effort required to check the execution of the described navigation rules.

For the reduction of these efforts, the following devices and functions are effective:

1. Simple description of the navigation rules, which does not require programming knowledge for the information providers.
2. Providing an authoring tool, which detects errors in the described navigation rules.

The above devices and functions are effective for the reduction of the information providers' effort in describing the

Table 3: Deadend in the navigation rule.

Subject	Task 1	2	3	4	5	6	Dead End ratio(%)
a	O	O	O	O	X	O	17
b	O	O	O	O	X	X	33
c	O	X	O	O	X	O	33
d	O	O	O	O	X	O	17
e	O	O	O	O	X	O	17
f	O	O	O	O	O	O	0
g	O	O	O	O	O	O	0
h	O	O	O	O	X	O	17
i	O	O	O	O	O	O	0
j	O	O	O	O	O	O	0

Table 4: Loop errors.

Subject	Task 5	6	Loop error ratio(%)
a	X	O	50
b	X	X	100
c	X	O	50
d	X	O	50
e	X	O	50
f	O	O	0
g	O	O	0
h	X	O	50
i	O	O	0
j	O	O	0

navigation rules and reducing the errors in the described navigation rules for general hypermedia systems where information providers want to guide users.

CONCLUSIONS

This paper proposed an adaptive hypermedia system that reduces information providers' efforts to describe the navigation rules and leads to fewer errors in the described navigation rules while guiding users accurately. This system uses simple expressions for the navigation rules to reduce the information providers' efforts. It also adapts the hyperspace to the user by link hiding in order to achieve the desired user paths. We also offer an authoring tool for this system, which checks whether there are errors in the described navigation rules, with the aim of further reducing the information providers' efforts to describe the navigation rules and avoid errors in those rules.

The proposed system was implemented and evaluated qualitatively and quantitatively. In the qualitative evaluation, five information providers freely described content and navigation rules and gave the experimenter their subjective opinions. In the quantitative evaluation, ten information providers described navigation rules for the same navigation tasks and the experimenter measured the time required for describing the navigation rules and the error ratio, which is the ratio of

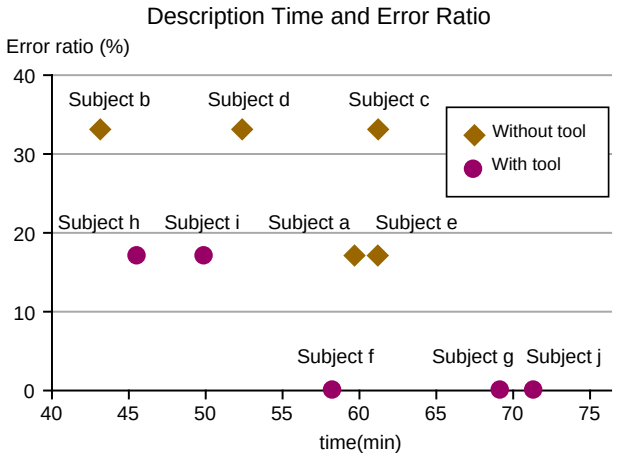

Figure 5: Time for describing and error ratio

the navigation tasks with errors in relation to all navigation tasks. The results of the experiments provide evidence supporting the effectiveness of the system in the reduction of information providers' efforts and in minimizing navigation errors. The proposed functions are effective for hypermedia systems in which information providers want to guide users properly.

Our future research will focus on an enhanced authoring tool and an enhanced rule function.

REFERENCES

1. Brusilovsky, P. L.: Methods and Techniques of Adaptive Hypermedia, User Modeling and User-Adapted Interaction, Vol.6, No.2-3, (1996), pp.87-129.
2. Brusilovsky, P. L.: Intelligent Tutor, Environment and Manual for Introductory Programming, Educational and Training Technology International, Vol.29, No.1, (1992), pp.26-34.
3. Brusilovsky, P. L., Eklund, J. and Schwarz E.: Web-based Education for All: A Tool for Development Adaptive Courseware, Computer Networks and ISDN Systems (Proc. of the 7th International World Wide Web Conference), Vol.30, (1998), pp.291-300.
4. Boyle, C. and Encarnacion, A. O.: MetaDoc: An Adaptive Hypertext Reading System, User Modeling and User-Adapted Interaction, Vol.4, No.1, (1994), pp.1-19.
5. De Bra, P. and Calvi, L.: AHA: a Generic Adaptive Hypermedia System, Proc. of 2nd Workshop on Adaptive Hypertext and Hypermedia (1998), http://wwwis.win.tue.nl/ah98/Proceedings.html.
6. De Bra, P., Houben, G. and Wu, H.: AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, Proc. of Hypertext'99 (1999), pp.147-156.
7. Chittaro, L. and Ranon, R.: Adding Adaptive Features to Virtual Reality Interfaces for E-Commerce, Interna-

tional Conference on Adaptive Hypermedia and Adaptive Web-based Systems, LNCS 1892 (2000), pp. 86-97.

8. Gonschorek, M. and Herzog, C.: Using Hypertext for an Adaptive Helpsystem in an Intelligent Tutoring System, Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education (1995), pp.274-281.
9. Hohl,H. et al.: Hypadapter: An Adaptive Hypertext System for Exploratory Learning and Programming, User Modeling and User-Adapted Interaction, Vol.6, No.2-3, (1996), pp.131-156.
10. Perez, T. et al.: HyperTutor: From Hypermedia to Intelligent Adaptive Hypermedia, ED-MEDIA'95 - World Conference on Educational Multimedia and Hypermedia (1995), pp.529-534.
11. Rety, J.: Structure Analysis for Hypertext with Conditional Linkage, Proc. of Hypertext'99 (1999), pp.135-136.
12. Rich, E.: Users Are Individuals: Individualizing User Models, International Journal of Man-Machine Studies, Vol.18, (1983), pp.199-214.
13. Wu, H, De Bra, P., Aerts, A. and Houben, G.: Adaptation Control in Adaptive Hypermedia Systems, Proc. of International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, LNCS 1892 (2000), pp. 250-259.
14. Yanagisawa, A. and Matsumoto, H.: Internet Value Chain Marketing, (1998), SCC

APPENDIX

Data for Experiment

We created educational content for learning computer science. As regards the size of content, there are 103 nodes with 177 links. Table 5 shows the meaning of the user parameters. Table 6 shows the meanings of the classes. Table 7 shows the contents of the task, the number of nodes where navigation rules should be defined, and the types of the rules. The abbreviations "nu", "gu", and "np" in Table 7 stand for the node user rules, the general user rules, and the node path rules.

Table 5: User parameters for the experiments.

User parameter	Meaning
1	The degree of interest in the area of networks
2	The degree of interest in the area of system development
3	The degree of interest in the area of hardware
4	The degree of interest in the area of operating systems
5	The degree of knowledge in the area of networks
6	The degree of knowledge in the area of system development
7	The degree of knowledge in the area of hardware
8	The degree of knowledge in the area of operating systems

Table 6: Classes for the experiments.

Class	Role
A	Offering a question
B	Offering a response when the user answers the question correctly
C	Offering a response when the user answers the question incorrectly
D	Offering an explanation
E	Topic-related class (networks)
F	Topic-related class (system development)
G	Topic-related class (hardware)
H	Topic-related class (operating systems)

Table 7: Tasks in the experiments.

Task	Contents	No. of node	Rule type
1	Hide links to the teaching materials that the user is not interested in. This is based on the degree of interest for the four areas.	1	gu
2	Provide questions first, then provide explanations for the user whose degree of knowledge is high. Provide explanations first, then provide questions for the user whose degree of knowledge is low.	6	nu
3	Provide three questions, then change the contents of the explanation according to the eight patterns that the users could answer.	2	np
4	Provide five questions, which are ordered from basic to difficult. After the user answers all questions, provide the same questions again beginning with the first question that the user answers incorrectly. If the user answers all questions correctly he/she is finished studying. However the user is only allowed to work on each question twice.	1	np
5	The user answers questions in three areas, which are hard disk, CPU, and memory, in this order. Each area provides two questions. If the user answers even one question in an area incorrectly, he/she has to answer the same two questions again for the topic. If the user answers both of the questions correctly, he/she goes forwards to the next area.	3	np