Title	A Study of Supporting Personalization in Information Browsing System
Author(s)	土方, 嘉徳
Citation	大阪大学, 2002, 博士論文
Version Type	VoR
URL	https://hdl.handle.net/11094/679
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

工学8680

Doctoral Dissertation

Title

A Study of Supporting Personalization
in Information Browsing System

Author

Yoshinori Hijikata

January 2002

Osaka University,
Graduate School of Engineering Science,
Department of Systems and Human Science

Doctoral Dissertation

Title

A Study of Supporting Personalization
in Information Browsing System

Author

Yoshinori Hijikata

January 2002

Osaka University,
Graduate School of Engineering Science,
Department of Systems and Human Science

Abstract

This doctoral dissertation describes some fundamental methods supporting the personalization on information browsing systems.

The spread of the Internet, the WWW (one of the information browsing systems) is becoming a de facto standard not only as an application for information gathering but also as a platform for running applications. The WWW is attracting people's attention from the viewpoint of both technology side and business side. Especially "personalization" is the most expected technology. Personalization is the provision to the individual of tailored products, services, information or GUI(Graphical User Interface).

This research focuses on the user information gathering function and the authoring function for personalization. The user information gathering function is the function to automatically acquire the information about the user's interest. The authoring function is the function to support the information provider to attach an index to information or to describe how to personalize the information according to each user.

For supporting the user information gathering function, we propose a method to extract a text part that the user was interested in based on the user's ordinary mouse operations while he/she is reading a Web page. For supporting the authoring function, we considered supporting methods from two features of the WWW: (i) it allows users to acquire information by moving from a page to another page by a link and (ii) it provides GUIs as a platform for building Web applications. To the former feature, the most popular personalization method is a rule-based approach, in which the information provider describes the navigation strategies in rules in advance. We propose a method supporting the information provider by verifying the rules. To the latter feature, the method to attach the help function to the content or the application is attracting people's attention. This is called an EPSS (Electronic Performance Support System). We propose a framework for building EPSS independent of the content.

This doctoral dissertation consists of 5 chapters. Chapter 1 explains an information browsing system and a personalization on it. It also describes the background of this

work. Chapter 2 introduces a system called TextExtractor, which extracts the target text part of the operation which seems to be related to the user's interest. It explains the preliminary survey on the user's operations, the system structure of TextExtractor and the evaluation of the extracted keywords. Chapter 3 describes a verifying tool of the user-navigation strategy on a information browsing system. It evaluates the tool in the description time and the error ratio of the described strategies. Chapter 4 introduces a system called WebAttendant, which is a content-independent framework for Web-based EPSS. It explains the system structure of WebAttendant and the experiment to evaluate it, which compares the amount of work to develop EPSS using WebAttendant with that using built-in EPSS. Chapter 5 gives the conclusions considering the results in each chapter.

Contents

1 Introduction 1

1.1 Background 1

1.2 Information Browsing System 2

1.3 Objective of Our Research 4

1.4 Organization of the Dissertation 6

2 TextExtractor: Text Part Extraction Using Operation Logs from a Web Browser 9

2.1 Introduction 9

2.2 Related Work 11

2.3 Preliminary Survey 13

2.4 Experimental Method and System Implementation 15

2.4.1 Experimental objective and method 15

2.4.2 Experimental system TextExtractor 16

2.4.3 Operation extraction and text extraction 17

2.5 Evaluation 18

2.5.1 User's browsing in the experiment 18

2.5.2 Objective of evaluation 19

2.5.3 Validity for type of operation 21

2.5.4 Comparison with other methods 24

2.5.5 Conclusion of the experiment 28

2.6 Discussion and Future Work 28

2.6.1 Discussion 28
2.6.2 Future work 30
2.7 Conclusions 31

3 Supporting Information Providers for Rule-based Adaptive Hypermedia 38
3.1 Introduction 38
3.2 Adaptation Method and User Model 40
3.2.1 Adaptation method 40
3.2.2 User model 41
3.3 Navigation Method 41
3.3.1 Hypermedia model 41
3.3.2 Class 42
3.3.3 Representation of user model 43
3.3.4 Navigation rule 43
3.3.5 Example of navigation 44
3.4 Authoring Tool 46
3.4.1 Objective 46
3.4.2 Dead end detection 46
3.4.3 Detection of dead ends caused by path rules 47
3.4.4 Detection of dead ends caused by user rules 48
3.4.5 Loop detection 49
3.5 Implementation and Evaluation 50
3.5.1 Implementation of the system 50
3.5.2 Objective of evaluation 51
3.5.3 Evaluation method 52
3.5.4 Evaluation result 53
3.5.5 Discussion 58
3.6 Conclusions 59

4 Content-Independent Framework for Web-based EPSS **66**

4.1 Introduction 66

4.2 Concept of EPSS and Related Works 68

4.2.1 Concept of EPSS 68

4.2.2 Related works 68

4.2.3 EPSS for Web sites 70

4.3 System Design and Implementation 71

4.3.1 Objective and design plan 71

4.3.2 Outline of the system 72

4.3.3 Modules of WebAttendant 74

4.4 Evaluation 81

4.4.1 Objectives of evaluation 81

4.4.2 Evaluation on an amount of work 81

4.4.3 Evaluation on reusability for different Web sites 83

4.4.4 Evaluation on reusability for different skill levels 84

4.4.5 Summary of the results 86

4.4.6 Future directions 86

4.5 Conclusions 89

5 Conclusions **91**

Contents of Figures

1.1 System structure of information gathering system. 3

2.1 User ratio according to the number of operations. 15
2.2 Questionnaire window. 16
2.3 System structure for the experiment. 17
2.4 Keyword precesion for each type of operation. 23
2.5 Number of performed operations per a page. 23
2.6 Keyword precision. 26
2.7 Keyword recall. 27
2.8 Noise recall. 27

3.1 Hypermedia model. 42
3.2 An example of the rules. 45
3.3 An example of dead end detection. 49
3.4 Output example from the system. 51
3.5 Time for describing and error ratio. 58

4.1 WebAttendant. 68
4.2 WebAttendant and existing EPSS. 69
4.3 System structure of WebAttendant. 73
4.4 Guidance example. 78
4.5 Rule format. 78
4.6 An example of rule expression in XML. 79
4.7 Authoring tool. 80

Contents of Tables

2.1 Related works. 13
2.2 Browsing data for the experience. 20
2.3 Keyword narrowing rate. 26
2.4 Difference among the types of operations. 29
2.5 Parameters for detecting operations. 37

3.1 Description time. 56
3.2 Error in the navigation rule. 56
3.3 Deadend in the navigation rule. 57
3.4 Loop errors. 57
3.5 User parameters for the experiments. 63
3.6 Classes for the experiments. 64
3.7 Tasks in the experiments. 65

4.1 Context event. 75
4.2 DOM action. 77
4.3 Scenario and number of functions(rules) of the created EPSS. 82
4.4 The amount of development. 87
4.5 Reusability. 88

Chapter 1

Introduction

1.1 Background

As the Internet has been expanding, the WWW (World-Wide Web)[1] has become one of the most popular application on the Internet. The WWW is the application which supports the user to find some information from computers on the Internet. The WWW has the hyperspace consisting of pages (nodes) and links. Users access the hyperspace using a Web browser on their client computers. They acquire information by moving from a page to another page by a link. In the WWW, the information provider can easily exhibit information just by starting a Web server on the computer connected with the Internet and putting the files which describe the contents of the pages on the server. Therefore the number of pages has been increasing and it reached one billion pages[2] in 2000. In this situation, it is difficult for users to find their target information on the WWW. Futhermore, the WWW not only provides a method to access information, but also provides GUIs (graphical user interface) such as inputs and buttons and a protocol to send a query message. This allows the user to access databases or other existing legacy applications through Web browser. Therefore the WWW has become not only a tool for information retrieval but also a platform for every application[3].

From such a background, personalization is beginning to attract attention[4]. Personalization is the provision to the individual of tailored products, services, informa-

tion or information relating to products or service[5]. For example, personalization (1) emphasizes the link in which the user is likely to be interested from his/her browsing history, (2) displays the goods on the top page of the Web site recommended to the user based on his/her purchase information or (3) displays how to input the form by balloon helps for a beginner user. The point that personalization differs from other technologies is taking into consideration both (1) the user's usability to support the user's information browsing or the user's usage of the application and (2) the information provider's business to provide adequate information to his/her customers. Therefore, we have to consider the technology supporting personalization from various sides.

1.2 Information Browsing System

There are three types of systems which support the user's information gathering activity:

1. Information retrieval
2. Information filtering
3. Information browsing

Some researchers call these systems together "information gathering system."[6, 7] Information retrieval supports the user in finding information from remote databases. This type of system asks the user to input implicitly some keywords related to the user's interest. Information filtering supports the user in screening the information coming into the user. This type of system acquires the information about the user's interest in advance, and after that it compares the information coming to the user with the information about the user's interest. Information browsing provides the information space where the user can freely move around and gather information. The related information is connected by links by an information provider based on his/her point of view. The fundamental function provided in the WWW is information browsing although information retrieval and information filtering can be implemented there.

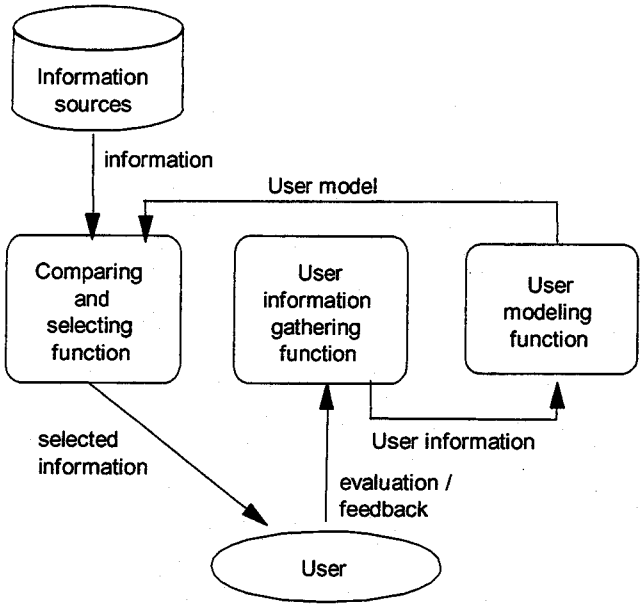

Figure 1.1: System structure of information gathering system.

Personalization needs the information about the user because it changes the content or the way to display the content according to each user. The basic system structure of information gathering systems using the information about the user is as follows[8, 9, 7] (see Figure 1.1):

1. User information gathering function
2. User modeling function
3. Comparing and selecting function

Users return some feedback to the system such as inputting keyword queries, expanding them and evaluating/judging information provided by the system. The information gathering function acquires those user information and sends them to the user modeling function. The user modeling function constructs or modifies the user model based on the user information provided by the user information gathering function.

The comparing and selecting function compares the user model with the information to provide the user. After that, it selects the information matched to the user's interest.

In addition to the above-mentioned functions, some systems need an authoring function. The comparing and selecting function compares the information with the user model and selects information to show the user. However this function often requires the computational presentation of the information and the strategy how to select the information. Even the information is presented in text, it is difficult to understand the meaning of the content by a computer. Therefore the information provider has to prepare computational presentations such as properties and values[10, 11]. Furthermore, the selecting and displaying strategy of information may differ for information providers. In this case, he/she has to define their way of selection and display in the system.

1.3 Objective of Our Research

In this research, we will study on the fundamental technologies required for the personalization on the information browsing system. Out of the four functions described in Section 1.2, this research will focus on the user information gathering function and the authoring function. Especially for the authoring function, we will pay attention to the two features of the WWW: (i) it allows users to acquire information by moving from a page to another page by a link and (ii) it provides GUIs as a platform for building Web applications. We will propose (1) TextExtractor as one the user information gathering functions, (2) a verifying tool for user-navigation as one of the authoring tools for the former feature of the WWW and (3) WebAttendant as one of the authoring tools for the latter feature of the WWW.

(1) TextExtractor

One of the serious requirements in information gathering systems is the high-accuracy acquisition of the user information without the user's mental effort. TextExtractor automatically extracts a text part that the user was interested in from the whole text of a page using the user's ordinary mouse operation. The idea of TextEx-

tractor is feasible not only for information browsing systems but also for information retrieval and information filtering systems.

(2) A verifying tool for user-navigation

The original idea of the WWW is hypermedia[12, 13], which is a basic idea of connecting information with links. Therefore hypermedia is one of the information browsing systems. In the research field of hypermedia, personalization has been studied as adaptive hypermedia[14]. Although some researchers have been studying on authoring tools for adaptive hypermedia systems, these tools do not focus on examining the navigation strategies in hyperspace. We will propose a verifying tool which examines the information provider's navigation strategies.

(3) WebAttendant

In the field of computer education and business applications, EPSS (Electronic Performance Support System)[15] is gaining its popularity. EPSS is like a help system for application software. However it differs from a help system in taking the user's business productivity into consideration. We will propose WebAttendant for EPSS. This is a framework for an EPSS on the Web and allows the EPSS developer to develop an EPSS independent of the Web content. Therefore WebAttendant can improve the reusability of the modules used in the EPSS and decrease the development cost.

In this research, TextExtractor is built as a module feasible in any Web site. For the verifying tool for user-navigation, we will develop a simple adaptive hypermedia system. We will develop the verifying tool on this system and see its effectiveness by an experiment. The reason why we will develop a whole system of the adaptive hypermedia is that we want to make the comparing and selecting mechanism and the user interface for the information provider simple. In the experiment this removes the mental workloads occured by such complexity from the information provider and see only the effectiveness of the tool. For WebAttendant, we will develop it as a whole system required for a standard platform of Web-based EPSS. This is because we want to see its effectiveness in the real EPSS development for business purpose.

1.4 Organization of the Dissertation

In this doctoral dissertation, we describe TextExtractor in Chapter 2, the verifying tool for user-navigation in Chapter 3 and WebAttendant in Chapter 4. In each chapter, we will describe the detail of their research backgrounds, target problems, methodologies, implementations and evaluations. In Chapter 5, we will give the conclusions about the supporting functions for personalization in information browsing systems.

Preferences

[1] Berners-Lee, T., et al.: The World-Wide Web, *Comm. of the ACM*, Vol. 37, No. 8, pp. 76–82 (1994).

[2] Broder, A., et al.: Graph Structure in the Web, *Proc. of the Ninth International World Wide Web Conference, In Computer Networks and ISDN Systems*, Vol. 33, pp. 309–320 (2000).

[3] Special Features: Software Service Technologies for Making e-Business Real, *IPSJ Magazine*, Vol. 42, No. 9, pp. 855–895 (2001).

[4] Special Features: Personalilzed Views of Personalization, *Comm. of the ACM*, Vol. 43, No. 8, pp. 27–158 (2000).

[5] Mulvenna, M.D., Anand, S.S., Buchner, A.G.: Personalization on the Net using Web Mining, *Comm. of the ACM*, Vol. 43, No. 8, pp. 123–125 (2000).

[6] Takeda, H.: Network Enhanced Intelligent Information Integration, *Journal of Japanese Society for Artificial Intelligence*, Vol. 11, No. 5, pp. 680–688 (1996).

[7] Sugimoto, M.: User Modeling and Adaptive Interaction in Information Gathering Systems, *Journal of Japanese Society for Artificial Intelligence*, Vol. 14, No. 1, pp. 25–32 (1999).

[8] Belkin, N. J. and Croftk, W. B.: Information Filtering and Information Retrieval: Two Sides of the Same Coin?, *Comm. of the ACM*, Vol. 35, No. 12, pp. 29–38 (1992).

[9] Loeb, S.: Architecting Personalized Delivary of Multimedia Information, *Comm. of the ACM*, Vol. 35, No. 12, pp. 39–48 (1992).

[10] Nagao, K.: Advanced Use of Digital Content Based on Annotation (Part 1), *IPSJ Magazine*, Vol. 42, No. 7, pp. 668–675 (2001).

[11] Nagao, K.: Advanced Use of Digital Content Based on Annotation (Part 2), *IPSJ Magazine*, Vol. 42, No. 8, pp. 787–792 (2001).

[12] Conklin, J.: Hypertext: An Introduction and Survey, in "Computer-Supported Cooperative Work," Morgan Kaufmann Publishers, pp. 423–475 (1988).

[13] Nielsen, J.: Hyper Text & Hyper Media, (1989), Academic Press.

[14] Brusilovsky, P. L.: Methods and Techniques of Adaptive Hypermedia, *User Modeling and User-Adapted Interaction*, Vol. 6, No. 2-3, pp. 87–129 (1996).

[15] Stevens, G.H. and Stevens, E.F.: Designing Electronic Performance Support Tools: Talent Requirements, *Performance & Instruction*, Vol. 24, No. 2, pp. 9–11 (1995).

Chapter 2

TextExtractor: Text Part Extraction Using Operation Logs from a Web Browser

2.1 Introduction

There are many search engine services on the Web that support users in acquiring their target information. When the user inputs some keywords as a search key, the search engine recommends pages that include the input keywords. The number of pages accessible by search engines has passed one billion pages[1]. Technologies for narrowing the number of search results are regarded as important, and many researchers have been working on these technologies. Methodology that relieves users from needing special knowledge about search engines on the Web is important, because there are many kinds of users on the Web. Relevance feedback[2] is one such method.

Relevance feedback (i) asks the user to indicate pages most relevant to his/her interests from the search results, and (ii) searches again using keywords specific to those selected pages. Generally, the selection of keywords is done from the pages returned as search results by the search engines, and this method selects new keywords from the complete text of those pages. Therefore this method has a problem in that not all the selected keywords have to do with the user's interests[5]. Another problem

is that it takes a lot of effort by the users to indicate suitable pages.

In this paper, as a solution for the first problem, we propose using only the parts that the user might be interested in, instead of using the entire pages. As a solution for the second problem, we propose using the user's browsing operations to determine his/her interests instead of asking the user to explicitly indicate the pages that the user had an interest in.

We focus on the situation that the user uses a mouse as an input device while browsing Web pages and solve the above-mentioned problems by the following method:

1. Extract operations that might occur because of the user's especial interest from the user's ordinary mouse operations while browsing pages.
2. Extract by sentence or line the text parts that are the targets of those extracted operations.

We expect the following results by using this method:

1. The system can automatically find the keywords relevant to the user's interests without requiring any special efforts by the user.
2. The system can eliminate many noise keywords, the keywords unrelated to the user's interests, from the texts used for relevance feedback.

This chapter is organized as follows. Section 2.2 introduces related work that gathers the information about users' interests, what we call the "user profile". Section 2.3 investigates the relationship between the users' interests and their mouse operations. Section 2.4 explains the experiment to see the effectiveness of the text parts which is the target of the operation which may relate to the user's interest. It also explains the system called "TextExtractor" which extracts the text parts based on the user's operation. Section 2.5 evaluates the effectiveness of the extracted keywords by TextExtractor. Section 2.6 provides some discussions and outlines future works. Finally Section 2.7 offers some conclusions.

2.2 Related Work

Relevance feedback acquires user profiles by asking the users to indicate which pages are relevant to their interests [2]. Some researchers have also been working on the acquisition of user profiles in the area of information filtering[3]. Information filtering screens the information coming into the user, while information retrieval supports the user in finding information from remote databases[4].

In the research areas of information retrieval and information filtering, there are two basic approaches for acquiring user profiles[5, 6].

1. Explicit (Direct) method:

   This method acquires user profiles by (i) asking users to answer preliminary questionnaires about topics or keywords which they are interested in, or (ii) asking users to grade the pages they have browsed for interest and relevance. Ringo[7] and SIFT[8] use the former approach. GroupLens[9], Syskill & Weber[10], NewsWeeder[11], ClixSmart[12] and AntWorld[13] use the latter approach. The advantage of this method is that it is reliable because it acquires the user profiles directly from the users. However these approaches also have some disadvantages. Generally, completing a preliminary questionnaire sufficiently detailed to allow a user to adequately describe his/her interests as keywords is a troublesome task, and grading pages also takes a lot of efforts from the users. Method (ii) also has a problem that it selects keywords from the whole text of the page and the selected keywords include many that the user is not interested in.

2. Implicit (Indirect) method:

   This method acquires user profiles by estimating the users' degree of interest in the pages the users have browsed based on such factors as (i) the time spent reading the pages (browsing time)[14] or (ii) the specific mouse button operations or the scroll operations performed while reading the pages[15], or (iii) the user's eye mark while reading pages[16, 17]. The advantage of this method is that it does not require any mental efforts by the users. One of the problems with method

(i) is that the system usually cannot know when the user opens a page and then starts doing some other work or leaves the PC. Existing research on method (ii) monitors for such actions as when the user pushes a button for enlarging an article in a news system or when the user scrolls the window that is displaying the article. Detecting these operations allows the system to judge whether the user was interested in the entire page. However the system cannot always locate which part of the page the user was interested in from these operations. Method (iii) has a possibility to specify the text part that the user was interested in. However it leaves the problem of the special equipment to recognize the user's eye mark.

In the research area of information retrieval and filtering, the system selects keywords effective for information retrieval or information filtering from the keywords in the documents that the user has rated as good ones. This keyword selection relies on tf-idf weights [2]. The tf-idf approach weights keywords based on each keyword's appearance frequency in the document and its appearance frequency in other documents. However tf-idf weights keywords based on the statistics of the entire document even if the user was only interested in a part of the document. Therefore some of the weights on keywords do not reflect the user's interests. Furthermore, the tf-idf algorithm needs to prepare a target document set and create a vector space for the set in advance, because it is based on considering relevance between documents.

Our research can be classified with the implicit methods because it estimates the users' interests based on the mouse operations. It differs from the existing research approaches in that (i) it estimates the user's interest from the ordinary mouse operations, including even the ones the user performs unconsciously, (ii) it extracts the parts the user might be interested in not by the page but by the sentence or line, and (iii) it does not need a special equipment (see Table 2.1).

Table 2.1: Related works.

Method	Category	Required for user's efforts?	Unit of extraction	Required for special equipment?	Works
Preliminary questionnaire	Explicit	Yes	Topic/keyword	No	Ringo, SIFT
Page rating	Explicit	Yes	Page	No	GroupLens, AntWorld Syskill & Weber NewsWeeder, ClixSmart
Browsing time	Implicit	No	Page	No	Morita
Special button & scroll operation	Implicit	No	Page	No	ANATAGONOMY
Eye mark	Implicit	No	Part of page	Yes	Digital Reminder IMPACT
TextExtractor	Implicit	No	Part of page	No	

2.3 Preliminary Survey

Kantor[13] reports that he discovered that users tend to follow the mouse pointer by the eye while browsing Web pages. As one of the reasons of above-mentioned behavior, he pointed out that the user has to click links that he/she is interested in by the mouse on the Web. However he does not show what kind of operations performed by users while browsing Web pages and whether or not such operations have to do with their interests.

We surveyed characteristic operations which may occur according to users' interests. In this survey, we conducted observations of and interviews about users' operations while they are browsing Web pages. In these observations, the users freely browsed Web pages they liked and the observer watched their mouse operations. In the interviews, the interviewer asked the users what kind of operations they perform in their daily Web browsing and the reason why they perform the operations. There were 31 users in the survey. This survey detected the following characteristic operations. (We eliminated operations to directly specify the targets of the users' interests such as inputting some

keywords that the user is interested in into the text field of a search engine.)

- Text tracing: Moving the mouse pointer along a sentence while reading.
- Link pointing: Positioning the mouse pointer on a link, but not clicking the link.
- Link clicking: Clicking on a link to move to another page.
- Text selection: Selecting text by dragging the mouse pointer.
- Scrolling: Scrolling a window at a certain speed.
- Bookmark registration: Registering a page as a bookmark.
- Saving: Saving an HTML document.
- Printing: Printing a page.
- Window movement: Moving a window of the Web browser.
- Window resizing: Changing the window size of the Web browser.

Some of the operations are necessary for browsing Web pages or using the Web browser's functions. The other operations are not necessary for browsing Web pages or using the Web browser's functions, but users perform them unconsciously. Out of these operations, the operations whose targets can be text are text tracing, link pointing, link clicking and text selection.

To judge whether or not we can use these four kinks of operations for extracting text parts, it is necessary to see how many users perform them. We observed the 20 users' operations during 10-minute browsing and counted the number of times that each operation occurred. Figure 2.1 shows the result. Although there is a variety in the number of times to perform according to the type of operation, we found that in every type of operation there are users who perform it. Therefore in this research, we will investigate whether or not the target text part of these four kinds of operations are actually the part the user was interested in by the experiment.

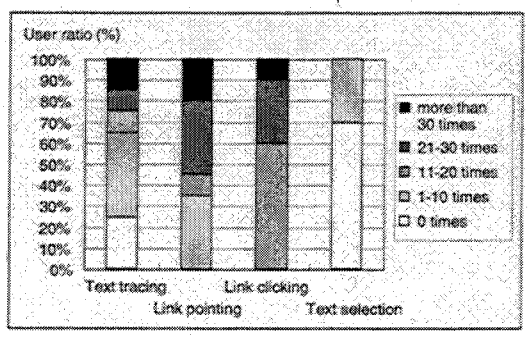

Figure 2.1: User ratio according to the number of operations.

2.4 Experimental Method and System Implementation

2.4.1 Experimental objective and method

Generally the unit of process in information retrieval and information filtering is keyword. The objective of the experiment is that automatically extracting the target text parts of the four types of operations (described in Section 2.3) and seeing whether or not keywords in the extracted text is actually the ones the user is interested in.

The experimental method we used is as follows:

1. The subject searches for the pages he/she wants to browse in advance of the experiment.
2. When the experimental observations begin, the subject freely browses the selected pages.
3. Every time the subject moves from a page to another page, he/she answers a questionnaire about the previous page. In this questionnaire, the system displays all keywords extracted from the page, and the subject checks only keywords he/she was interested in (see Figure 2.2).
4. The experimenter compares the keywords checked by the subject and the key-

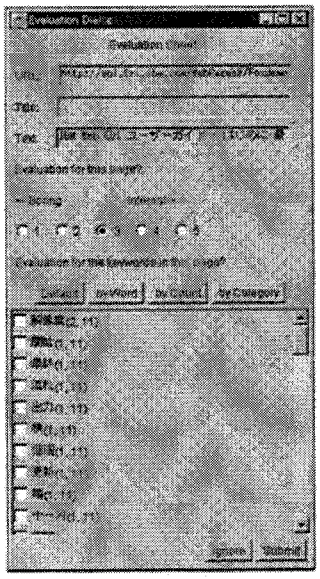

Figure 2.2: Questionnaire window.

words extracted by the system. The experimenter determines the effectiveness by calculating some parameters.

2.4.2 Experimental system TextExtractor

We developed a system called "TextExtractor" which extracts the text parts the user might be interested in while the user is browsing a Web page. We built TextExtractor using JavaScript and Java. This is because the user can use the Web browser that he/she usually uses and it is easy to install TextExtractor to exisiting Web contents and applications.

Figure 2.3 shows the system structure of TextExtractor. For embedding the JavaScript and Java applet programs of TextExtractor into Web pages, we developed an embed proxy server. In this system, the embed proxy server calls a morphological analyzer[18, 19] and generates windows for questionnaires using keywords output by a morphological analyzer. TextExtractor sends the extracted text parts to the TextEx-

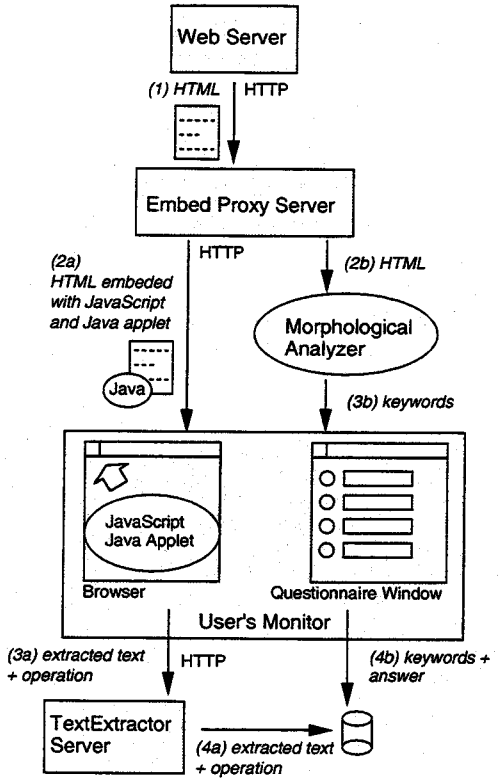

Figure 2.3: System structure for the experiment.

tractor server.

2.4.3 Operation extraction and text extraction

JavaScript program of TextExtractor detects the user's operation event on the Web browser via DOM(Document Object Model)[20] interface. After that, it informs Java applet program the event with other parameters such as coordinates of the mouse pointer in a fixed format. Java applet program extracts the four kinds of operations by analyzing the operation events and extracts the target text part of the operation. We show (1) the type of operation event, (2) the format to inform operation events, (3) an example described in the format, (4) concrete methods to extract operations,

(5) concrete methods to extract text parts and (6) parameters for operation extraction in the appendix.

We use some parameters for extracting operations. We set the parameters heuristically by recording five users' operations while they are browsing Web pages and analyzing them (see Appendix). In actual text tracing operation found in the preliminary experiment in Section 2.3 and in the operation recording in this section, users do not strictly trace the line they are reading but just unconsciously move the mouse pointer to the right and in short distance. Therefore we set parameters to make TextExtractor recognize the short mouse movement to the right as text tracing.

We use some functions of DynamicHTML[21] for extracting the text. When the user selects a text part with mouse, the Web browser generates a selection object, which indicates the selected text. We use this object for extracting the text which is the target of the text selection operation. We can also identify the location of a character which is at specific coordinates from the whole text of the page. We use this function for extracting the text which is the target of text tracing, link clicking, and link pointing operations. When the system extracts text which is the target of a text tracing operation, it also extracts the text which exists on the line above the line where the mouse pointer is. This is because of what we discovered from the observations. We found two cases in users' text tracing operations: (i) they move the mouse pointer on a straight line within the line where they are reading, or (ii) they move the mouse pointer on a line below the text line where they are reading.

2.5 Evaluation

2.5.1 User's browsing in the experiment

Five users (three women and two men in their twenties or thirties) participated in the experiment as subjects. We used data from 120 Web pages for the analysis. Table 2.2 shows the objective of each user's browsing, the characteristics of the pages that each user browsed (also showing the average number of keywords in those pages), the

average number of keywords the user checked as interesting ones in each page and the number of pages the user has browsed.

2.5.2 Objective of evaluation

In this section, we will see whether or not the target text part of each type of operation is actually the part the user was interested in. Namely we will see whether or not the ratio of keywords that the user was interested in is higher in the target text part of each type of operation than in the whole text of the page. After that, we will compare TextExtractor, which extracts keywords based on the four kinds of operations, with other keyword extraction method. In this comparison, we will see the validity in text extraction of TextExtractor by comparing TextExtractor with the method to extract keywords at random, we call this random extraction. We also compare TextExtractor with tf-idf which is the most popular keyword selection method in information retrieval and information filtering.

In this evaluation we will calculate the following three parameters:

1. Keyword precision
2. Keyword recall
3. Noise recall

Keyword precision is the ratio of the keywords that the user is interested in in relation to the the extracted keywords. Keyword recall is the ratio of the extracted keywords in relation to the keywords that the user is interested in. This parameter is important because in information retrieval or information filtering, the system cannot recommend the pages the user is interested in when the number of keywords to use, which the user is interested in, is small. Noise recall is the ratio of the extracted keywords in relation to the keywords that the user is not interested in (noise keywords). This parameter is important because the noise keywords reduce precision of information retrieval when the system executes relevance feedback using keywords in the whole text

Table 2.2: Browsing data for the experience.

User	The objective of browsing	The characteristics of the pages (Average number of keywords in a page)	NC	NP
User A	Clicking a link of the mail magazine published by a news site on the mail software and browsing each news article.	Pages consisting of text and some figures. Figures are banner ads and photos for the article. (198)	3.8	20
User B	Browsing a Web site for cars from its top page. Clicking a link of the mail magazine published by a news site on the mail software and browsing each news article.	A top page with many links and pages consisting of text and some figures. Figures are banner ads. (351)	3.6	20
User C	Browsing personal sites from their top pages for essays and restaurant information.	Top pages with many links and pages consisting of text and some figures for the articles. (241)	4.7	29
User D	Selecting Web sites for a popular singer in a commercial index service site and browsing concert information and bulletin boards in each site. Clicking a link of the mail magazine published by a news site on the mail software and browsing each news article.	A page with many links in an index service site, each site's top page with some links and figures, pages offering data as lists or tables, and pages of bulletin board. Few figures except for each site's top page. (157)	1.1	25
User E	Browsing some personal or cities' Web sites offering travel information.	Top pages with some links and pages consisting of text and some figures. Some of the figures are large maps. (124)	2.7	26

NC: Average number of keywords checked by the user as interesting ones in a page.
NP: Number of pages the user browsed.

of the page. When we subtract noise recall from 1, we get the ratio of the reduced noise keywords in relation to the noise keywords in the page (noise reduction rate).

Considering the usage in information retrieval and information filtering, keyword precision can evaluate the effectiveness of the extracted keywords. Using keyword recall and noise recall besides keyword precision, we can evaluate the effectiveness of the keyword extraction method. The equations to calculate these parameters are as follows:

1. Keyword precision $=\mid B \mid / \mid A \mid$
2. Keyword recall $=\mid D \mid / \mid C \mid$
3. Noise recall $=\mid F \mid / \mid E \mid$

A, B, C, D, E and F in the above equations have the following meanings:

- A: The set of extracted keywords.
- B: The set of keywords which are included in the set A and checked by the user as interesting ones.
- C: The set of keywords checked by the user as interesting ones in the whole text of the page.
- D: The set of keywords which are included in the set C and extracted by TextExtractor.
- E: The set of noise keywords in the whole text of the page.
- F: The set of noise keywords which are included in the set E and are not extracted by TextExtractor.

2.5.3 Validity for type of operation

Figure 2.4 shows the keyword precision in every type of operation and the keyword precision in the whole text of the page. The keyword precision is higher in the extrated text than in the whole text for every user and for every type of operation.

Figure 2.5 shows the number of times the user performed each type operation in a page. We can see there is individual difference in the frequency to perform the operation in text tracing and link pointing operation. The frequency to perform link click of User A is lower than other users. This is because User A read each new article by clicking the links of a mail magazine published by a news site on his mail software and hardly clicked links on the Web page.

Although there was individual difference depending on the type of operation, we saw the extracted text part in every type of operation includes the keywords that the user was interested in at higher ratio than in the whole text of the page.

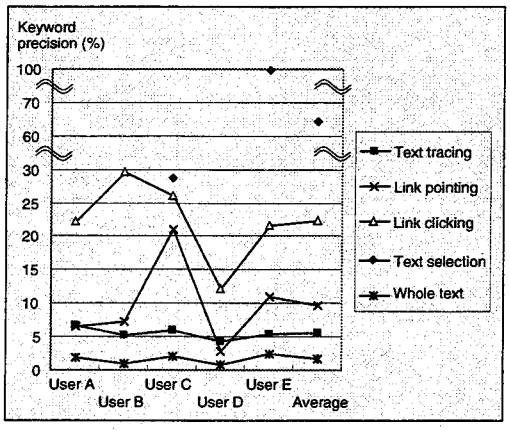

Figure 2.4: Keyword precesion for each type of operation.

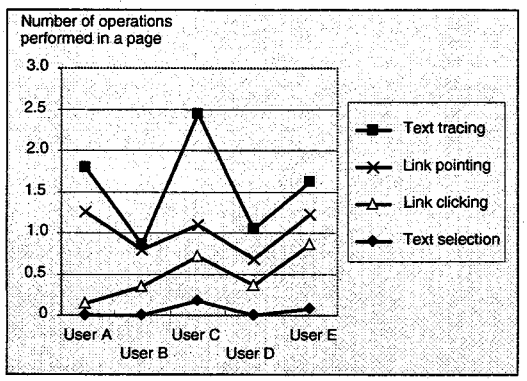

Figure 2.5: Number of performed operations per a page.

2.5.4 Comparison with other methods

Random extraction and tf-idf can extract keywords at any ratio from the whole text of the page. However TextExtractor cannot extract keywords at any ratio. Therefore we will calculate keyword narrowing rate which represents how much TextExtractor narrows the text part from the whole text of the page. We will extract keywords at keyword narrowing rate of TextExtractor in random extraction and tf-idf. This means we will compare these methods when they extract keywords at the same ratio. The equation to calculate keyword narrowing rate is as follows:

Keyword narrowing rate $=\mid H \mid / \mid G \mid$

H and G in the above equation have the following meanings:

- G: The set of keywords in the whole text of the page.
- H: The set of keywords which are included in the set G and extracted by TextExtractor.

In random extraction, we calculated the expectation of keyword precision, keyword recall and noise recall when we extract keywords from the whole text of the page. The expectation of keyword precision is keyword precision in the whole text of the page browsed by the user. The expectation of keyword recall and noise recall is the keyword narrowing rate of TextExtractor. tf-idf needs the document set defined in advance. In this experiment, we created vector spaces using keywords in all pages browsed by each user. We selected keywords based on the weights of tf-idf at the keyword narrowing rate.

Table 2.3 shows the keyword narrowing rate. Figure 2.6, 2.7, 2.8 shows the keyword precision, the keyword recall and the noise recall. Compared to random extraction, the keyword precision and the keyword recall of TextExtractor is approximately four times on the average of all users. The difference of the noise recall between TextExtractor and random extraction is small although the noise recall of TextExtractor is slightly better than random extraction. This is because more than 98% of keywords in the whole text of the page is noise keywords (We can see this from that the average keyword precision

of all users in the whole text of the page is less than 2%). Therefore TextExtractor extracts more keywords the user was interested in than random extraction and reduces noise keywords at almost the same ratio of random extraction.

Compared to tf-idf, the keyword precision and the keyword recall of TextExtractor is about 1.4 times on the average of all users. For the users except for User A, the keyword precision and the keyword recall of TextExtractor is better than those of tf-idf. There is difference in browsing behaviors between User A and User B-E. User A browsed only the pages that display one news article about IT (Information Technology). User B-E browsed various kinds of pages such as top pages of Web sites, pages with link collections, pages of bulletin boards, pages with a personal diary and pages displaying some data in a table. tf-idf is a powerful method for documents consisting of many sentences such as news articles because it weighs keywords based on their frequency in the document. The pages browsed by User A are all news articles and include many sentences. Therefore the keyword precision and keyword recall of tf-idf has become high in those pages. The pages browsed by User B-E did not always include many sentences. Therefore the keyword precision and keyword recall of tf-idf has become low in those pages. Meanwhile TextExtractor extracts keywords based on the user's mouse operation and does not consider the keywords' frequency in the documents. Therefore it extracts the keywords that the user was interested in from the browsed pages, even if those pages do not include many sentences. This shows that TextExtractor can extract the keywords that the user was interested in at high accuracy even in various kinds of pages where tf-idf cannot achieve its best performance.

Table 2.3: Keyword narrowing rate.

User	Keyword narrowing rate(%)
User A	9.74
User B	3.50
User C	8.62
User D	7.76
User E	14.32
Average	8.78

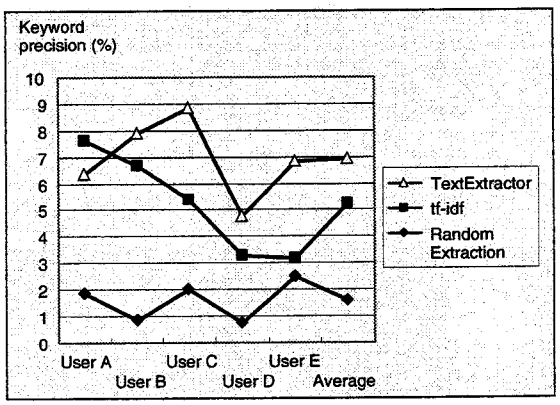

Figure 2.6: Keyword precision.

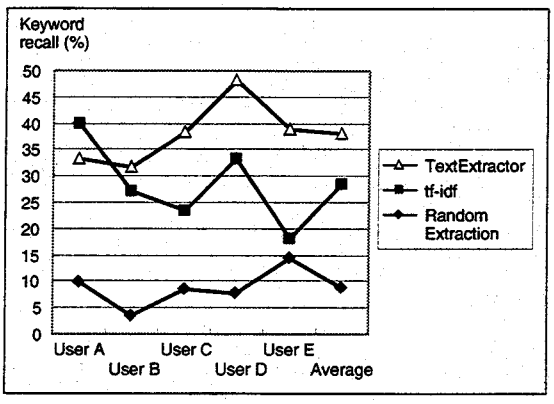

Figure 2.7: Keyword recall.

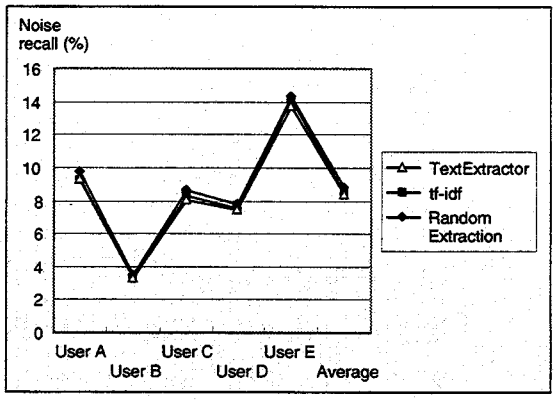

Figure 2.8: Noise recall.

2.5.5 Conclusion of the experiment

We confirmed that the target text part of text tracing, link pointing, link clicking and text selection operation includes keywords the user was interested in at higher accuracy than the whole text of the page. When we used these all four kinds of operations for keyword extraction in TextExtractor, it could extract keywords that the user was interested in at high accuracy even for pages with miscellaneous styles where tf-idf cannot achieve its best performance. Therefore we can expect a better Web page search by using keywords extracted by TextExtractor for relevance feedback.

In the experiment, five users browsed their favorite Web pages as they usually do. The result shows that TextExtractor extracts keywords that the user was interested in at high accuracy just by using the user's usual mouse operations. From this, we confirmed that the system can acquire the information about the user's interest without insisting users to answer the questionnaire about their interest or to grade the pages they have browsed. Therefore we can expect that TextExtractor allows users to use the functions supporting a Web page search such as relevance feedback more easily without inputting keywords or rating pages consciously.

2.6 Discussion and Future Work

2.6.1 Discussion

This subsection discusses the type of operation and the expiry of the user profile.

Type of Operation

The experiment in Section 2.5 calculated the keyword precision, the keyword recall and the noise recall for each user using all kinds of operations. This section will compare the four kinds of operations. Table 2.4 shows the keyword precision, the keyword recall and the noise recall in every type of operation considering all users. The keyword precision of operations performed unconsciously such as text tracing and link pointing is lower than that of operations performed consciously such as text selection and link

Table 2.4: Difference among the types of operations.

Type of operation	Keyword precision(%)	Keyword recall(%)	noise recall(%)
Text tracing	5.83	20.91	5.21
Link pointing	10.15	8.85	1.21
Link clicking	22.76	15.01	0.79
Text selection	50.00	1.34	0.02

clicking. There is difference in the keyword recall according to the type of operation. We can see that the keyword recall is not going to be high by using only operations with high keyword precision such as text selection and link clicking.

When the system searches Web pages using a search query with a few keywords, operations with high keyword precision such as text selection and link clicking will be effective. When the system searches Web pages using vector space models with many keywords, operations with high keyword recall such as text tracing and link pointing will also be effective. In this case, the system can also change the weights of the keywords according to the type of operation used for the keyword extraction. It will be important to select the type of operation for the keyword extraction and to weighs keywords based on the type of operation according to the target application.

Expiry of user profile

Some researchers work on the expiry of the user profile. Miyahara hypothesizes that the strength of the user's interest follows the Gamma distribution and tries to prove its correctness[22]. NewsT uses the genetic algorithm and leaves the genes with the current and strong interest[23]. SIFTER considers the history of relevance feedback (interesting or not on a category) as a Bernoulli trial and judges the change of occurrence probability by Bayesian analysis[24]. Crabtree categorizes browsed documents into fixed categories for a period of time and sees the differences between some periods[25]. IndexNavigator tries to infer the change of the user's interest by the hypothetical inference[26].

TextExtractor has a feature to infer the user's interest by a part of the page not by

a page. When the system stores the extracted text for a long period, there will be little difference between TextExtractor and a method to infer the user's interest by a page. Therefore we should use the extracted text parts for supporting the user's ongoing information acquisition. In this meaning, the expiry of the user profile extracted by TextExtractor is during its session.

2.6.2 Future work

The evaluation in Section 2.5 considers whether or not the user was interested in keywords. However the important characteristics of TextExtractor is that it extracts the text part that the user was interested in by a sentence or a line. When the system not only acquires keywords from the text but also analyzes the context of the text and the text in front and rear of it, the system may know how the user was interested in the topic. For example, when we consider the keyword 'Java', the context is different between "The market of Java applications has become big." and "The technology of Java applications has become improved." although they use the same keyword. The user who was interested in the former sentence, he/she may be interested in marketing. The user who was interested in the latter sentence, he/she may be interested in technology. It is important as a future work to analyze the context of the extracted text using methods natural language processing and apply it in the user analysis for information retrieval and for marketing.

As a text part extraction method using a user's behavior, there is a method using the user's eye mark. Although this method has a problem that it needs a special equipment to detect the user's eye mark, we think it is valuable to investigate its performance of text extraction. This is because it may acquire the user's interest which was not shown in the user's mouse operation. It is important as a future work to compare the method using the user's eye mark with random extraction, tf-idf and TextExtractor.

The representation type of information with which TextExtractor deals is text. However multimedia data has also become important. When we try to apply TextExtractor for multimedia data, the application (browser) must include some mouse

operations on the content while the user is watching or browsing it. However the problem is that such an application rarely exists now. Moreover even if it exists, there is another problem about the unit of extraction, i.e., it is difficult to define the meaningful part of the content.

2.7 Conclusions

This chapter describes a method to extract a text part which the user might be interested in using the user's mouse operation performed during his/her usual Web browsing. In our research, we conducted a preliminary survey and discovered four kinds of operations related to the users' interests: text tracing, link pointing, link clicking and text selection. We developed a system called "TextExtractor" which extracts the target text part of these four kinds of operations by sentence or line. We conducted an experiment to see if the extracted text by TextExtractor is actually the part the user was interested in.

Five users participated in the experiment and browsed their favorite pages as usual. The result shows that the target text parts of every four kinds of operations include keywords the user was interested in at higher ratio than whole text of the page. Comparing TextExtractor with the method to extract keywords at random, we confirmed that TextExtractor extract keywords that the user was interested in at about 4 times of accuracy. These results shows that TextExtractor extract the text part that the user was interested in without insisting users to answer questionnaires.

We also compared TextExtractor with tf-idf which is the most popular keyword selection method. The result shows that TextExtractor extracted the keyword that the user was interested in at about 1.4 times of accuracy. The result also showed that TextExtractor extract keywords at high accuracy even for pages with miscellaneous styles such as bulletin boards and link collections where tf-idf does not achieve its best performance. Therefore we can expect a more sophisticated information retrieval using the extracted text by TextExtractor for relevance feedback.

Our future research will use the keywords extracted by TextExtractor and see its

effectiveness. We will also analyze the context of the extracted text and apply it to the user analysis in information retrieval and marketing.

Preferences

[1] Broder, A., et al.: Graph Structure in the Web, *Proc. of the Ninth International World Wide Web Conference, In Computer Networks and ISDN Systems*, Vol. 33, pp. 309–320 (2000).

[2] Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison Wesley (1989).

[3] Resnick, P., et al.: Recommender Systems, *Comm. of the ACM*, Vol. 40, No. 3, pp. 56–89 (1997).

[4] Belkin, N. J. and Croftk, W. B.: Information Filtering and Information Retrieval: Two Sides of the Same Coin?, *Comm. of the ACM*, Vol. 35, No. 12, pp. 29–38 (1992).

[5] Sugimoto, M.: User Modeling and Adaptive Interaction in Information Gathering Systems, *Journal of Japanese Society for Artificial Intelligence*, Vol. 14, No. 1, pp. 25–32 (1999).

[6] Mulvenna, M.D., Anand, S.S., Buchner, A.G.: Personalization on the Net using Web Mining, *Comm. of the ACM*, Vol. 43, No. 8, pp. 123–125 (2000).

[7] Shardanand, U. and Maes, P.: Social Information Filtering: Algorithm for Automating 'Word of Mouth', *Proc. of CHI'95*, pp. 210–217 (1995).

[8] Yan, T. W. and Garcia-Molina, H.: SIFT - A Tool for Wide-Area Information Dissemination, *Proc. of 1995 USENIX Technical Conference*, pp. 177–186 (1995).

[9] Resnick, P., et al.: GroupLens : An Open Architecture for Collaborative Filtering of Netnews, *Proc. of CSCW'94*, pp. 175–186 (1994).

[10] Pazzani, M. and Billsus, D.: Learning and Revising User Profiles: the Identification of Interesting Web Sites, *Machine Learning*, Vol. 27, No. 3, pp. 313–331 (1997).

[11] Lang, K.: NewsWeeder: Learning to Filter NetNews, *Proc. of ICML'95*, pp. 331–339 (1994).

[12] Smyth, B. and Cotter, P.: A Personalized Television Listings Service, *Comm. of the ACM*, Vol. 43, No. 8, pp. 107–111 (2000).

[13] Kantor, P.B., et. al: Capturing Human Intelligence in the Net, *Comm. of the ACM*, Vol. 43, No. 8, pp. 112–115 (2000).

[14] Morita, M. and Shinoda, Y.: Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval, *Proc. of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 272–281 (1994).

[15] Sakagami, H. and Kamba, T.: Learning Personal Preferences on Online Newspaper Articles from User Behaviors, *Proc. of the Sixth International World Wide Web Conference, In Computer Networks and ISDN Systems*, Vol. 29, pp. 1447–1456 (1997).

[16] Yoshida, M. and Yoshitaka, A.: Digital Reminder : Building and Accessing a Real-World-Oriented Database, *Proc. of The 8th Workshop on Interactive Systems and Software (WISS'2000)*, pp. 101–110 (2000).

[17] Ohno, T.: IMPACT : Eye Mark Reusing Technique to Support Information Browsing Task, *Proc. of The 8th Workshop on Interactive Systems and Software (WISS'2000)*, pp. 137–146 (2000).

[18] Okumura, M.: Introduction of Natural Language Processing Tools, *IPSJ Magazine*, Vol. 41, No. 11, pp. 1203–1207 (2000).

[19] Matsumoto, Y.: Morphological Analysis System ChaSen, *IPSJ Magazine*, Vol. 41, No. 11, pp. 1208–1214 (2000).

[20] http://www.w3.org/DOM/

[21] Goodman, D.: Dynamic HTML The Definitive Reference, O'Reilly (1998).

[22] Miyahara, K. and Okamoto, T.: Quantified Estimation Method of User's Information Interests based on the Web Browsing and its Application to Collaborative Filtering, *Proc. of the Technical Report of IEICE*, ET97-115, pp. 17–24 (1998).

[23] Shetch, B. and Maes, P.: Evolving Agents for Personalized Information Filtering, *Proc. of IEEE Conference on Artificial Intelligence for Applications*, pp. 345–352 (1993).

[24] Mostafa, J., et al.: A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation, *ACM Transactions of Information Systems*, Vol. 15, No. 4, pp. 368–399 (1997).

[25] Crabtree, I.B. and Soltysiak, S.J.: Identifying and Trackign Changing Interests, *Iternational Journal of Digital Library*, Vol. 4, pp. 38–53 (1998).

[26] Osawa, Y., Sugawa, A. and Yachida, M.: An Index Navigator:Underatanding and Expressing User's Changing Interest, *Journal of Japanese Society for Artificial Intelligence*, Vol. 13, No. 3, pp. 461–469 (1998).

[27] Hijikata, Y., Aoki, Y., Furui, Y. and Nakajima, A.: TextExtractor: Text Part Extraction Using Operation Logs on Web Browser, *Proc. of The 8th Workshop on Interactive Systems and Software (WISS 2000)*, pp. 201–206 (2000).

Appendix

Operation event

1. Focus: A focus event occurs when the user clicks an object and the object becomes ready to receive input data from the keyboard.
2. Blur: A blur event occurs when the user clicks an object and the current object loses the focus.
3. Mousemove: A mousemove event occurs when the user moves the mouse pointer.
4. Mousedown: A mousedown event occurs when the user presses the mouse button.
5. Mouseup: A mouseup event occurs when the user releases the mouse button.
6. Resize: A resize event occurs when the user resizes the window.
7. Scroll: A scroll event occurs when the user scrolls a window.
8. Click: A click event occurs when the user clicks on an object.
9. Mouseover: A mouseover event occurs when the user moves the mouse pointer over an object.
10. Mouseout: A mouseout event occurs when the user moves the mouse pointer off of an object.
11. Select: A select event occurs when the user selects text by dragging the mouse pointer.

Format of operation event

operation time, operation event type, frame ID where the event occurred, object type involved in the event, and other data such as the coordinates

Examples of operation event

936332393593,blur,frames(0),7,BODY
936332407468,focus,frames(0),7,BODY
936332410218,mouseover,frames(0),7,BODY,215,0
936332410265,mousemove,frames(0),7,BODY,215,0

Operation extraction method

1. Text tracing
   First, the system detects continuous movement of the mouse pointer in a horizontal direction. For this detection, every time a mousemove event occurs, the system calculates the angle of the mouse movement relative to the horizontal and the time between the current mousemove event and the previous mousemove event.

(For calculating the angle, the system uses the current mousemove event, and a mousemove event that occurred n-times before.) If the angle is below a threshold Ar and the time is below a threshold Tr, the system regards the movement as a continuous movement in a horizontal direction.
Second, when the system detects such a movement of the mouse pointer, the system calculates the distance and velocity of the movement. If the distance is longer than a threshold L and the velocity is slower than a threshold V, the system regards this operation as a text tracing operation.

2. Link pointing
   The system regards the operation as a link pointing operation when a mouseover event occurs on a link object, but there is no click event afterwards, and a mouseout event occurs after a time Tp.
3. Link clicking
   The system regards a click event on a link object as a link clicking operation.
4. Text selection
   The system regards the operation as a text selection when a mouseup event at the end of a select event occurs.

Table 2.5: Parameters for detecting operations.

Parameter	Value
Angle $A_r(\mid \tan\theta \mid)$	0.25
Time T_r	750(msec)
Length L	40(pixels)
Velocity V	0.45(pixels/msec)
History n	2
Time T_p	750(msec)

Chapter 3

Supporting Information Providers for Rule-based Adaptive Hypermedia

3.1 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 [1]. 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 [2]. 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 [3]. Adaptive hypermedia is hypermedia [4, 5] 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 [6]–[22]. 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.

Most existing rule-based adaptive hypermedia systems [6]–[17],[19],[22] do not focus on providing mechanisms and functions that can reduce the burden on information providers for creating and verifying navigation rules. (Rety[18] uses Prologue and Stotts[20] uses Lisp as a rule language. In their systems, information providers can use general tools for these programming languages to create and verify rules. However the format of these programming languages is complex for information providers, and these tools just verify the grammatical errors in the rules but do not verify the correctness of the navigation.)

Some researchers have been studying on authoring tools for adaptive hypermedia systems [21, 23, 24, 25, 26]. However these systems do not offer the function examining the navigation rules. (Wadge's system[21] only proposes the markup language for authoring. InterBook[23] allows information providers to use popular word processing applications (MS-Word) and converts the created file to the format for InterBook. However it is not a rule-based adaptive hypermedia system and it does not allow information providers to direct users. Petrelli's system[24] focuses on the design of hypermedia network, and ECSAIWeb[25] and NetCoach[26] focuses on defining and modifying the knowledge concept used for adaptation. However these systems do not support information providers to describe 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 [27], and (3) the format of user information is also simple, because the format of navigation rule is simple.

This chapter 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.

3.2 Adaptation Method and User Model

Existing adaptive hypermedia systems construct a user model and use it for adapting to each user [3]. 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.

3.2.1 Adaptation method

Adaptation methods can be classified into content-level adaptation and link-level adaptation[3]. 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 and adaptive annotation) according to how the links are modified [3]. (Brusilovsky also mentioned map adaptation in addition to the above mentioned methods. Maps usually graphically represent a hyperspace or local area of hyperspace as a network of nodes connected by arrows [3]. However this representation is different from ordinary documents of hypermedia.) 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.

3.2.2 User model

As user models for adaptive hypermedia, there are overlay models and stereotype models [3]. Models using keywords are also used in some adaptive hypermedia systems [28, 29, 30]. 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.

Since it is generally important to consider long-term and short-term user information for designing user models [27], 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.

3.3 Navigation Method

3.3.1 Hypermedia model

Figure 3.1 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

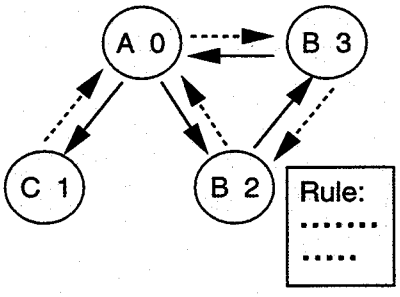

Figure 3.1: Hypermedia model.

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.

3.3.2 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).

3.3.3 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 $CABBA$.

3.3.4 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

$$C_{11} \cdots C_{1h} + \cdots + C_{m1} \cdots C_{mh} = D_1, \cdots, D_n \tag{3.1}$$

2. General path rule

$$C_{11} \cdots C_{1h} + \cdots + C_{m1} \cdots C_{mh} = C_1^f, \cdots, C_n^f \tag{3.2}$$

3. Node user rule

$$e_1 \# P_i \# e_2 : D_1, D_2, \cdots, D_n \tag{3.3}$$

4. General user rule

$$e_1 \# P_i \# e_2 : C_1^f, C_2^f, \cdots, C_n^f \tag{3.4}$$

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.

3.3.5 Example of navigation

Figure 3.2 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.

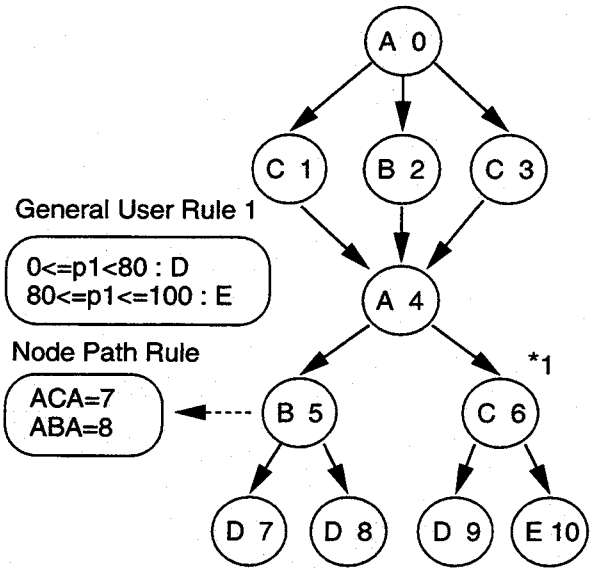

Figure 3.2: An example of the rules.

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 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.

3.4 Authoring Tool

3.4.1 Objective

Generally an authoring tool is important for an adaptive hypermedia system so that the information provider can direct users in hyperspace [8, 9, 14, 18, 21, 23]. 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 "getting lost" problems [4, 5] for user navigation.

3.4.2 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.

3.4.3 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 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 3.3 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.

3.4.4 Detection of dead ends 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:

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

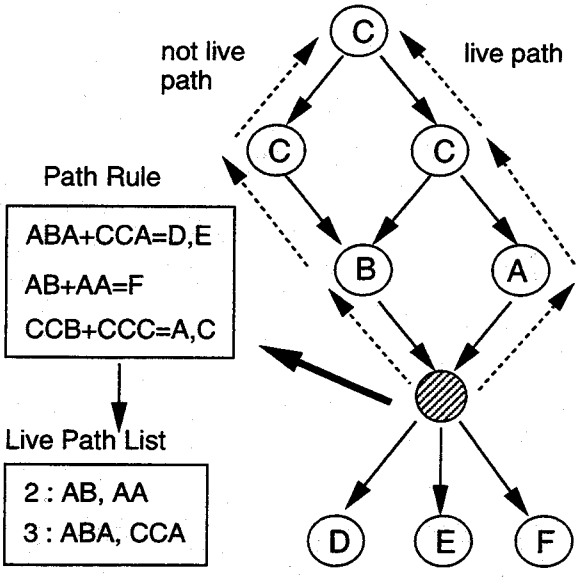

Figure 3.3: An example of dead end detection.

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.

3.4.5 Loop detection

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 loops 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.

3.5 Implementation and Evaluation

3.5.1 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 3.4-(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 3.4-(b,c) shows examples of the authoring tool. Figure 3.4-(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

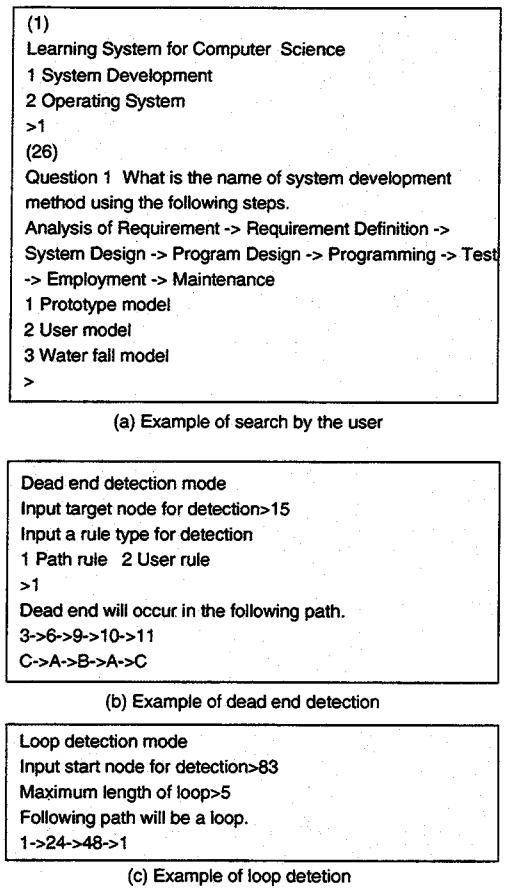

Figure 3.4: Output example from the system.

and causing dead end. Figure 3.4-(c) is an example of loop detection. It shows the sequences of the node IDs of the detected loop.

3.5.2 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.

3.5.3 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.

3.5.4 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 3.1,3.2,3.3,3.4, Subjects a-e were in Group A and Subjects f-j were in Group B. Table 3.1 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 3.2 shows whether or not the subject described the navigation rules without an error, and the error ratio. Table 3.3 shows whether or not the navigation rule that the subject described included dead ends, and the dead end ratio. Table 3.4 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 3.5 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

Table 3.1: 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 3.2: 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

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 3.3: 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 3.4: 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

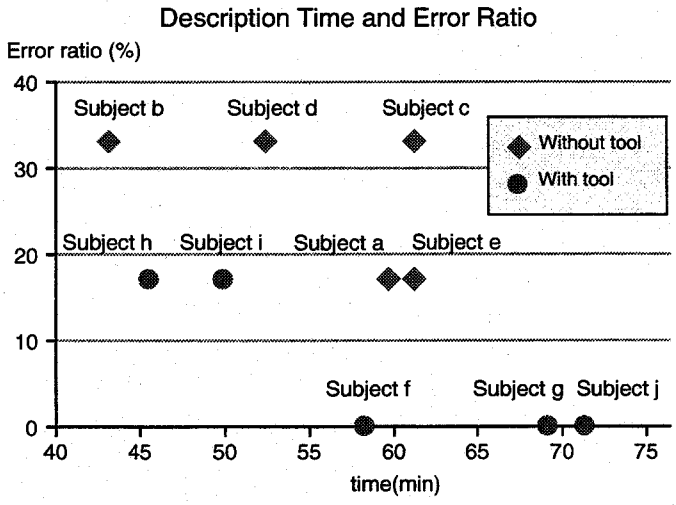

Figure 3.5: Time for describing and error ratio.

3.5.5 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 navigation rules and reducing the errors in the de-

scribed navigation rules for general hypermedia systems where information providers want to guide users.

3.6 Conclusions

This chapter 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 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.

Preferences

[1] Yanagisawa, A. and Matsumoto, H.: Internet Value Chain Marketing, (1998), SCC

[2] Miller, M. and Wantz, L. J.: Computed Web Links: The COOL Link Model, *Proc. of 2nd Workshop on Adaptive Hypertext and Hypermedia*, (1998), http://wwwis.win.tue.nl/ah98/Proceedings.html.

[3] Brusilovsky, P. L.: Methods and Techniques of Adaptive Hypermedia, *User Modeling and User-Adapted Interaction*, Vol. 6, No. 2-3, pp. 87–129 (1996).

[4] Conklin, J.: Hypertext: An Introduction and Survey, in "Computer-Supported Cooperative Work," Morgan Kaufmann Publishers, pp. 423–475 (1988).

[5] Nielsen, J.: Hyper Text & Hyper Media, (1989), Academic Press.

[6] Brusilovsky, P. L.: Intelligent Tutor, Environment and Manual for Introductory Programming, *Educational and Training Technology International*, Vol. 29, No. 1, pp. 26–34 (1992).

[7] Boyle, C. and Encarnacion, A. O.: MetaDoc: An Adaptive Hypertext Reading System, *User Modeling and User-Adapted Interaction*, Vol. 4, No. 1, pp. 1–19 (1994).

[8] 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.

[9] De Bra, P., Houben, G. and Wu, H.: AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, *Proc. of ACM Hypertext'99*, pp. 147–156 (1999).

[10] De Carolis, B. and Pizzutilo, S.: From Discourse to User-Adapted Hypermedia, *Proc. of UM'97, International Conference on User Modeling*, pp. 37–40 (1997).

[11] Chittaro, L. and Ranon, R.: Adding Adaptive Features to Virtual Reality Interfaces for E-Commerce, *Proc. of AH2000, International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, LNCS 1892, pp. 86–97 (2000).

[12] Fischer, G. et al.: Minimalist Explanations in Knowledge-based Systems, *Proc. of 23-th Annual Hawaii International Conference on System Sciences*, pp. 309–317 (1990).

[13] 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*, pp. 274–281 (1995).

[14] 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, pp. 131–156 (1996).

[15] Kay, J. and Kummerfeld, R. J.: An Individualised Course for the C Programming Language, *Proc. of Second International WWW Conference*, pp. 17–20 (1994).

[16] Not, E. and Zancanaro, M.: The MacroNode Approach: Mediating Between Adaptive and Dynamic Hypermedia, *Proc. of AH2000, International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, LNCS 1892, pp. 167–178 (2000).

[17] Perez, T. et al.: HyperTutor: From Hypermedia to Intelligent Adaptive Hypermedia, *Proc. of ED-MEDIA'95 - World Conference on Educational Multimedia and Hypermedia*, pp. 529–534 (1995).

[18] Rety, J.: Structure Analysis for Hypertext with Conditional Linkage, *Proc. of ACM Hypertext'99*, pp. 135–136 (1999).

[19] De Rosis, F., De Carolis, B. and Pizzutilo, S.: User Tailored Hypermedia Explanations, *INTERCHI'93 Adjust Proceedings*, pp. 169–170 (1993).

[20] Stotts, P. and Furuta, R.: Dynamic Adaptation of Hypertext Structure, *Proc. of Third ACM Conference on Hypertext*, pp. 219–231 (1991).

[21] Wadge, W.W. and Schraefel, M.C.: A Complementary Approach for Adaptive and Adaptable Hypermedia: Intensional Hypertext, *Proc. of 3rd Workshop on Adaptive Hypertext and Hypermedia*, (2001), http://wwwis.win.tue.nl/ah2001/proceedings.html.

[22] Wu, H, De Bra, P., Aerts, A. and Houben, G.: Adaptation Control in Adaptive Hypermedia Systems, *Proc. of AH2000, International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, LNCS 1892, pp. 250–259 (2000).

[23] 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, pp. 291–300 (1998).

[24] Petrelli, D., Baggio, D. and Pezzulo, G.: Adaptive Hypertext Design Environments: Putting Principles into Practice, *Proc. of AH2000, International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, LNCS 1892, pp. 202–213 (2000).

[25] Sanrach, C. and Grandbastien, M.: ECSAIWeb: A Web-Based Authoring System to Create Adaptive Learning Systems, *Proc. of AH2000, International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, LNCS 1892, pp. 214–226 (2000).

[26] Weber, G., Kuhl, H. and Weibelzahl, S.: Developing Adaptive Internet Based Courses with the Authoring System NetCoach : Intensional Hypertext, *Proc. of 3rd Workshop on Adaptive Hypertext and Hypermedia*, (2001), http://wwwis.win.tue.nl/ah2001/proceedings.html.

[27] Rich, E.: Users Are Individuals: Individualizing User Models, *International Journal of Man-Machine Studies*, Vol. 18, pp. 199–214 (1983).

[28] Armstrong, R., et al.: WebWatcher: A Learning Appretice for the World-Wide Web, *Proc. of 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, pp. 6–12 (1995).

[29] Joachims, T., Freitag, D., and Mitchell, T.: WebWatcher: A Tour Guide for The World Wide Web, *Proc. of the 15th International Joint Conference on Artificial Intelligence*, pp. 770–775 (1997).

[30] Lieberman, H.: Letizia: An Agent that Assists Web Browsing. *Proc. of International Joint Conference on Arttificial Intelligence*, (1995), http://lieber.www.media.mit.edu/people/lieber/Lieberary/Letizia/Letizia-AAAI/Letizia.html

[31] Hijikata, Y., Yoshida, T. and Nishida, S.: Adaptive Hypermedia System for Supporting Information Providers to Direct Users through Hyperspace, *Transaction of IEEJ*, Vol. 120-C, No. 11, pp. 1720–1731 (2000).

[32] Hijikata, Y., Yoshida, T. and Nishida, S.: Adaptive Hypermedia System for Supporting Information Providers in Directing Users through Hyperspace, *Proc. of Third Workshop on Adaptive Hypertext and Hypermedia*, pp. 147–156 (2001).

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 3.5 shows the meaning of the user parameters. Table 3.6 shows the meanings of the classes. Table 3.7 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 3.7 stand for the node user rules, the general user rules, and the node path rules.

Table 3.5: 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 3.6: 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 3.7: Tasks in the experiments.

Task	Contents	Number of nodes	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 more advanced contents for the users who answered correctly all of the three questions or whose degree of knowledge is high.	1	np up
5	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
6	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

Chapter 4

Content-Independent Framework for Web-based EPSS

4.1 Introduction

Use of Internet for things like online shopping or online reservations has drastically increased in the recent years and continues to grow everyday. People carrying out these online activities need to complete some tasks on the Web sites. For example, they need to login or register on the Web sites, search the Web sites, click on right links and submit or complete their task by clicking on the appropriate buttons. Web sites of these types are considered "task-oriented Web sites".

We believe that there are two problems with existing "task-oriented Web sites":

- If the Web content is not organized well structurally, the Web site users may not easily understand how to use the Web site,
- If the users are not skilled enough to use Web, they will have significant difficulty using the task-oriented Web sites.

If users face any of the above problems they may repeatedly scroll up and down in order to find their way to what they are planning to accomplish on a Web page. Or, if they think they accessed the wrong page unrelated to their task, they may go back to the previous page trying to get to the link that may lead them to the target page. Unnecessary scrolling up and down and moving into and out of task-unrelated pages may result in a longer time for a given task completion.

One solution to the problems that task-oriented Web sites are facing is attempting to add an Electronic Performance Support System (EPSS[1]) to the Web site. Web-based

EPSS is designed using methods to modify the composition of the Web content. For example, by embedding help functions in the content and adding detailed explanations for the content on the Web page. Almost every existing Web-based EPSS is a "built-in EPSS, i.e., they have been developed for specific Web content and the service is part of the content. There are some problems with built-in EPSS:

- It is difficult to reuse a specific Web site's built-in EPSS for another Web content because the composition of any specific Web content is different from another Web content.
- Various versions of a built-in EPSS are required for a specific Web site to support users with various skill levels. Development and maintenance of various versions of a built-in EPSS is costly and time consuming.

To resolve the shortcomings of the built-in EPSS, we propose a framework of a content-independent EPSS on the Web called WebAttendant. As shown in Figure 4.1, WebAttendant is built independent of the Web content. WebAttendant automatically tracks the user's operation event on the Web page, which is acquired through DOM (Document Object Model)[2] interface, analyzes user's operation logs, and provides users with specific instructions. Instructions are in the form of helpful interventions by guidance window(s) beside the Web page and balloon help message pointing to the target object on a Web page. The tactics how to provide users with instructions are defined as rules by EPSS developers with the authoring tool.

The advantages of WebAttendant are following:

- WebAttendant's EPSS can be reused for Web sites with different contents and for different skill-level users just by changing the rules. This will reduce the cost for development and the maintenance of the whole system.
- EPSS developers can easily provide EPSS on the Web content only by creating rules using WebAttendant's authoring function.
- Since each process of WebAttendant is independent of the Web server, EPSS developers can add EPSS to the existing Web content without changing the Web content itself.

In order to verify the usefulness and efficacy of WebAttendant, we carried out several experiments. Results of the experiments show that WebAttendant is a highly effective platform for Web-based EPSS.

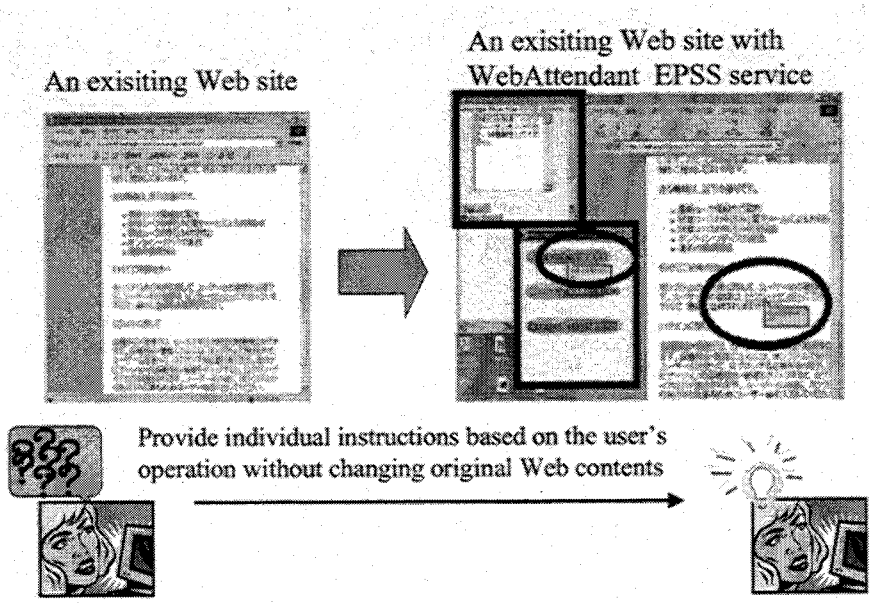

Figure 4.1: WebAttendant.

4.2 Concept of EPSS and Related Works

4.2.1 Concept of EPSS

Here we summarize the purpose of EPSS described by Stevens et al. [1]. An EPSS is a system that can provide on-demand, task-specific skills training, task- and situation-specific information access, customized tools for task automation, and embedded coaching, help, and validation tools. The aim of EPSS is to address the various levels of skill among the users of applications, in other words, to develop an EPSS that does not require its users to be necessarily highly skilled in the operation of the application.

4.2.2 Related works

Figure 4.2 shows WebAttendant and different categories of existing EPSS. Existing EPSS and WebAttendant are categorized based on their target applications. There are EPSS which are used for stand-alone applications and for Web sites.

EPSS for stand-alone applications

Context sense method	Explicit query	Automatic context sense
	CoachWare TRACK Knowledge base	Microsoft Agent MMhelper QuickCards

EPSS for Web sites

Context sense method / Implementation	Explicit query	Automatic context sense
Content-independent EPSS (For any Web site)		*WebAttendant*
Built-in EPSS (For specific Web site)	Query-based Online Help	Movie Help EPSS for the American FactFinder

Figure 4.2: WebAttendant and existing EPSS.

EPSS for standard-alone applications

EPSS for stand-alone applications are categorized based on the techniques they use to figure out what their users are trying to do. There are two categories of EPSS for stand-alone applications:

- EPSS using explicit query approach.
- EPSS using automatic context sensing approach

For the explicit query approach, users have to request support from the EPSS by sending explicit queries about what they are trying to do or about what help they want. CoachWare [3] and the TRACK Knowledge Base are examples of systems that use explicit queries to find out how to support a user's task. Such systems may use a database of queries that are intended to force users to clarify their situation even when users are not knowledgeable about the application they are trying to use. Alternately,

users may be asked to describe their problems in their own words. However, this approach is especially difficult for those users who don't understand the application.

For the automatic context sensing approach, the system attempts to infer each user's intentions by automatically tracking their operation histories. If a user makes a mistake, the system can detect his/her mistake and support him/her without any request for assistance being made by the user, even when users do not recognize that they have made mistakes. For this reason, an EPSS with an automatic context sensing approach is more user-friendly and superior to an EPSS with an explicit query approach. MMHelper [4], Microsoft Agent[5], and QuickCard[6] are examples of EPSS with the same approach. They attempt to automatically sense the context. Examples of useful data that can be used for this detection include the font size and number words input in a form, the amount of time a user spends in an application without doing anything. These existing EPSS detect the operations within the standard GUI components. Since these existing EPSS are designed for a stand-alone application, they cannot be reused for other applications.

4.2.3 EPSS for Web sites

Categorized from the viewpoint of system design, there are built-in EPSS and content-independent EPSS on the Web. A built-in EPSS is designed for specific Web content by building the EPSS as a part of the content. On the other hand, a content-independent EPSS is designed independently from any Web contents. Almost all existing Web-based EPSS are built-in EPSS. The same as EPSS for standalone applications, there are two categories of EPSS for Web sites:

- EPSS using explicit query approach.
- EPSS using automatic context sensing approach

Most existing EPSS on the Web are built-in EPSS with explicit query approach. The main examples of built-in EPSS with explicit query approach are Query-Based Online Help Services. In a Query-Based Online Help Service, a user has to ask questions by entering keywords or forming a query on the Web page and then get guidance from an EPSS database. A Query-Based Online Help Service does not assist a user unless he/she recognizes the mistakes.

Examples of existing built-in EPSS using automatic context sensing are a Movie Help Service on a Web site and the EPSS for the American FactFinder[7, 8]. The Movie Help Service demonstrates to the user how to use the Web site using balloon help

messages when a user loads the Web site. The Movie Help Service only tracks user's page loading events. It does not grasp user's intention. The EPSS for the American FactFinder provides performance support based on their unique characteristics and the needs of the key user group to which they belong. However, it mainly tracks user's page loading events, and it does not track more detailed information, as does WebAttendant. Plus, it is not reusable.

Because Web masters can design many kinds of interventions and make programs freely, built-in EPSS allows a greater flexibility in design. However, there are some disadvantages with built-in EPSS:

- If Web masters introduce the EPSS function on the existing Web site, they have to modify the content.
- It is difficult to reuse the modules of one built-in EPSS for other Web contents.
- If a large number of users with different skill levels use the Web sites, Web masters will need to develop several versions of a built-in EPSS service to support its users.

WebAttendant is content-independent EPSS using automatic context-sensing. The former characteristics of WebAttendant makes the development and testing processes much easier, and the development costs lower adding or reusing services without changing the existing Web content itself. The latter characteristics is achieved by grasping a user's status for offering instructions by automatically tracking the user's more detailed Web operations, such as, the time and the type of the operation event occurs, the type, number, and value of any target objects where the operation event occurs, the user name, and the current URL involved. WebAttendant can infer a user's intention more accurately and can provide the user with more individualized instructions than existing built-in EPSS using automatic context sensing. Considering the above-mentioned advantages of WebAttendant, it seems reasonable to consider it is superior to other EPSS.

4.3 System Design and Implementation

4.3.1 Objective and design plan

Our objectives for designing WebAttendant are to meet two requirements:

1. To create a framework for a cost-efficiency development of an EPSS

2. To develop an EPSS that does not require its users to be necessarily highly skilled in the operation of the application.

To achieve our first objective, we designed the WebAttendant to be a separate EPSS from the existing Web content. Also, WebAttendant separates the rule from the WebAttendant execution module to reuse the same rules for other Web content. Furthermore, WebAttendant provides an authoring tool so that EPSS developers (rule creators) can create rules easily, even if they do not have programming skills. To achieve our second objective, we designed WebAttendant to provide individual instructions based on the user's Web operation events with an automatic context sensing approach.

4.3.2 Outline of the system

Separation of the Web site and WebAttendant

Figure 4.3 shows the structure of WebAttendant. WebAttendant is designed to operate separately from a Web site. In the client side, the proxy server embeds several WebAttendant modules to track user's Web operation and provide him/her with interventions on the Web page. In the WebAttendant server side, other processes of WebAttendant, such as recording and analysis of user's operation and execution of the rules, operate separately from the Web site. As a result, WebAttendant services can be added, changed, and reused without a Web master changing the existing Web content. Also, Web sites and WebAttendant can be maintained individually.

The rules are separated from the WebAttendant execution module, allowing other Web contents to reuse the same rules for an EPSS and decreasing the number of rules that have to be created. WebAttendant provides an authoring tool to create rules easily.

Providing each user with individual instructions on a Web page

To provide each user with an individual instruction on a Web site, a method to recognize the user's status or intention by inference from the user's detailed operation history is required. Therefore, WebAttendant is required to provide a history tracking function, which automatically detects the following user's operation as many as possible on the Web site.

(1) Operations performed on several Web pages

Generally, a Web site consists of many pages. If the tasks in the Web site are complicated, users will need to carry out various operations over many pages. Therefore, a function, which tracks the users operations for several pages is required.

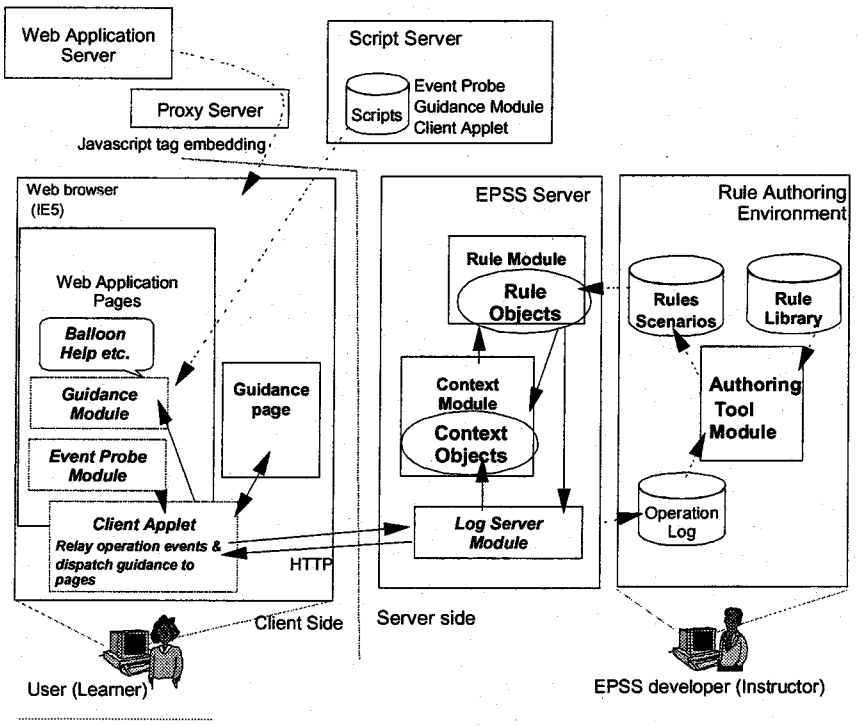

Figure 4.3: System structure of WebAttendant.

(2) Operations at each object in a page.

If the users' tasks in the Web site are complicated, they will need to carry out various operations in a page. WebAttendant detects the user's operations at each object (for example, each inputting form) in a page in order to examine if he/she carries out the operation in a correct way.

To analyze a user's operation, WebAttendant records and manages the user's operation as several contexts with names and attributes in the server side. Then, WebAttendant analyzes each user's contexts by examining some attributes to find out whether or not it has satisfied certain conditions. For example, it will examine to find out if the number of scrolling operations is less or more than a specific times. Then, it will examine the rules that need to be executed.

After WebAttendant determines to execute a rule to provide a user with individual instructions, it customizes the Web page dynamically by displaying several interventions at suitable places, or it modifies the Web page automatically in order to complete

a task efficiently.

4.3.3 Modules of WebAttendant

WebAttendant consists of following modules:

(1) Proxy server module

Proxy server module embeds the tags for event probe module and guidance module into each Web page. The tags refer to modules on the Script Server (a standard HTTP server). The Web page where the tags are embedded is then sent to a client browser [9].

(2) Event probe module

When a user operates in the Web page, the event probe module automatically tracks the user's operation by handling a DOM event in the Web browser [10]. Examples of a DOM events are CLICK, MOUSEMOVE, MOUSEOVER, SCROLL, KEYDOWN, KEYUP, LOAD, SELECT, and so on. The event probe module collects a complete set of information, including

- The time and the type of the DOM event that occurs,
- The type, number, and value of any target objects where the DOM event occurs,
- The IP address of the client or the user name,
- The current URL involved.

The collected information about the user's operations is sent to a log server.

(3) Log server module

Log server module receives the operation events from the event probe module and sends them to the context module. Log server module also transmits DOM actions (actions of interventions) from a rule module to the guidance module.

(4) Context module

Context module has general contexts and represents what a user has done. The context consists of a set of more than one context items with a name and attribute. There are three kinds of context items:

1. User context item: A parameter related to a user such as "current page's URL", "browsing history of URL" and "user's skill level."
2. Page context item: A parameter related to a page such as "URL", "browsing time" and "number of mouse movement."

Table 4.1: Context event.

Context Event	Meaning
load	Load a page
unload	Unload a page
mouseover	Put the mouse over the target object
mouseout	Put the mouse out of the target object
click	Click the mouse at the target object
submit	Submit the target form
no-change	Lose focus without inputting
repeated-focus	Focus the target object repeatedly
specified-focus-order	Focus some objects with specified-order
specified-focus-unorder	Do not focus some objects with specified-order
long-focus	Foucus an object for a long time
long-no-keypress	Focus an object for a long time without inputting
many-mouse-operations	Move a mouse many times
many-scroll-operations	Scroll many times
many-mouseover	Put a mouse over some objects many times
many-back-button-return	Back to previous page many times
click-hesitation	Hesitate a click operation
input-include-prohibited-char	Input characters with prohibited char

3. Object context item: A parameter related to an object such as "value", "number of focuses" and "number of change."

When the context module accepts the operation event from the log server, it makes a new context item, which is appropriate to a new user, a new page and a new object, or updates the value of the existing context item. Also, context module examines a value of a context item and if the value is satisfactory, according to the set standard parameter, it makes the context event to notify the rule module that user's context updated and sends it to the rule module. Table 4.1 shows the type of context events currently implemented and used in the evaluation (explained in the section 4.4).

(5) Rule module

Rule module decides which rule should be selected to provide individual instructions based on the user's context events and executes the selected rules. The rule module receives the context event from the context module. Then, the rule module (1) selects a rule that is appropriate to the context event, (2) checks the context described in the

condition part of the rule, and (3) decides the execution of a guidance to the user as a DOM action or the update of the value of context item as a context action. The rule module sends DOM action to the log server, which is a command for guidance module to execute guidance. Or, the rule module sends the context action to the context module, which is a command for the context module to update the value of a context item.

(6) Guidance module

Guidance module runs as a client-side applet and give an individual instruction to a user. When the guidance module receives a DOM action from the log server, it executes the DOM action, such as an action to provide interventions to the target object or change pages automatically[11]. Table 4.2 shows the type of DOM actions. Figure 4.4 shows an example of some DOM actions: "balloon help on"(Figure 4.4-a), "auto input"(Figure 4.4-b), "window on"(Figure 4.4-c) and "web page on"(Figure 4.4-d).

(7) Rule

Rule defines the functions of EPSS service, i.e., the way to instruct each user depending on the user's situation. Rules describe what kind of intervention should happen and how that should happen, depending on the user's situation. Rules are described as a XML rule format. A rule format consists of a "condition part" and an "execution part". In condition part, the user's situation is described by using context event and the attribute of a specific context item. In execution part, the way to instruct each user or the update of his/her contexts are described by using more than one DOM actions and context actions. Figure 4.5 shows the rule format. Figure 4.6 shows an example of the rule in XML format.

(8) Authoring tool module

Authoring tool module provides a simple point-and-click user interface to create rules. Authoring tool module allows users to create rules by minimum operations, linking them directly to the target object on the Web page. The EPSS developer can describe which URL and object, when, what kinds of messages, and how he/she want to show by the interventions on the Web site by selecting an item in the window for the authoring tool beside the target Web page (see Figure 4.7).

Table 4.2: DOM action.

DOM action	Function
page change	Change URL automatically from the current URL in a same window
window on	Create a new window and display a specific URL
balloon help on	Add a balloon help beside a HTML element with a message
balloon help off	Delete a displayed balloon help
wizard	Display a window as a wizard for a user
web page on	Display a specific URL Web page besides a HTML object in a Web page
web page off	Delete an inserted Web page
auto input	Input a message automatically in an input form
alert	Display an alert window
disable	Disable a specific input form
enable	Eable a specific input form
focus	Focus a specific input form automatically
invisible	Make invisible a specific object
visible	Make visible a specific object

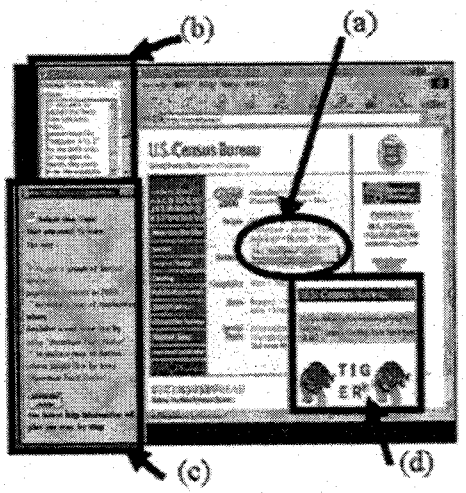

Figure 4.4: Guidance example.

source (condition part)	target (execution part)
+ context source + context event name + context condition clause + context descriptor + context condition	- DOM target + DOM object descriptor + DOM action descriptor - context target + context descriptor + context action descriptor

+ : consists of ...
- : is one of ...

Figure 4.5: Rule format.

```
<?xml version="1.0" ?>
<!DOCTYPE rule-set SYSTEM "rule.dtd">
<rule-set>
<rule>
    <source type="context">
        <context-event-name> long-focus </context-event-name>
        <context-condition-clause>
            <context-descriptor>
                <context-name>current-DOM-ID</context-name>
            </context-descriptor>
            <context-condition type="numeric">
                <operator>=</operator>
                <value>36</value>
            </context-condition>
        </context-condition-clause>
    </source>
    <target type="DOM">
        <DOM-object-descriptor>
            <object-identifier typer="ID">
                <DOM-ID>36</DOM-ID>
            </object-identifier>
        </DOM-object-descriptor>
        <DOM-action-descriptor type="param">
            <DOM-action-type>balloon-help-on</DOM-action-type>
            <DOM-action-parameter>
                <string-value type="constant">Input keywords here.</string-value>
            </DOM-action-parameter>
        </DOM-action-descriptor>
    </target>
</rule>
</rule-set>
```

Figure 4.6: An example of rule expression in XML.

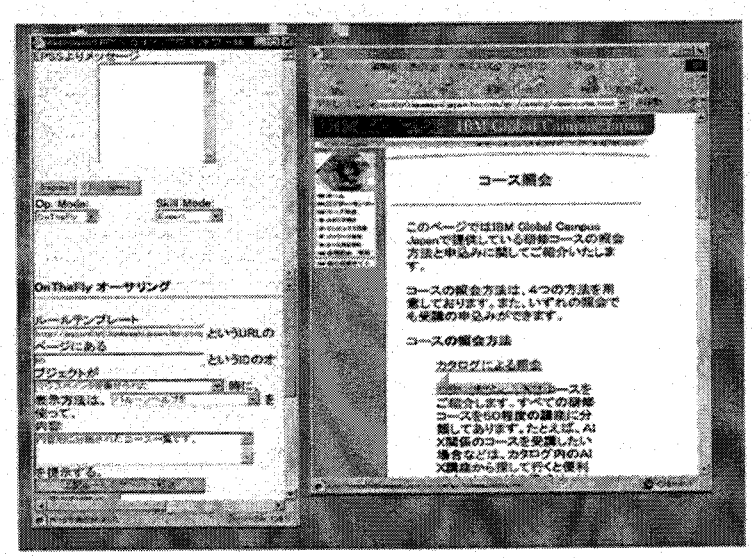

Figure 4.7: Authoring tool.

4.4 Evaluation

4.4.1 Objectives of evaluation

In order to verify the superiority of WebAttendant framework over a built-in EPSS in cost-efficiency of development, we ran several sets of experiments to evaluate two aspects of EPSS development:

(1) An amount of work involved to develop an EPSS

First we calculated the total operational steps involved in developing an EPSS when we used WebAttendant. Then, we examined the total size of the programs involved to develop the same EPSS when we used a built-in EPSS framework. Finally, We compared the total operational steps involved in EPSS development with WebAttendant framework to the total size of program involved in EPSS development with built-in EPSS framework.

(2) Reusability

To examine a cost-efficiency of development of an EPSS when EPSS modules are reused, we compared the total number of operational steps involved when we used WebAttendant framework to the size of programs involved when we used a built-in EPSS framework to reuse the same EPSS modules. We did this comparison for two possibilities:

(a) Developing EPSS for Different Web sites:

We developed several different EPSS for different Web sites. We then calculated the amount of work when modules for EPSS functions to a specific Web site were reused for other Web sites as well.

(b) Developing EPSS for users with different skill levels in a same Web site:

We developed several different EPSS for different skill levels in a same Web site. We then calculated the amount of work when modules for EPSS functions to a specific skill level were reused for other skill levels as well.

4.4.2 Evaluation on an amount of work

To compare an amount of work using WebAttendant framework versus using a built-in EPSS framework, we developed an EPSS called EPSS-A, which is for an existing Web training site. The scenario and the number of functions (rules) of EPSS-A is shown in Table 4.3.

Table 4.3: Scenario and number of functions(rules) of the created EPSS.

EPSS name	Scenario	Number of functions
EPSS-A	Assisting users how to register using their ID, check the availability of the courses, and fill out course application.	43
EPSS-B	Assisting users how to make a map using data in the US Census Bureau.	35
EPSS-C	Assisting users how to transfer a money to other's bank account.	30
EPSS-L1	The target is a user who does not have a Web operation skill, and also a first time user of this site. Assisting a user by explaining the flow of a task for his/her purpose in detailAnd also explaining the summary of each page.	10
EPSS-L2	The target is a user who have an ordinary Web operation skill and has used this site before. Assisting a user by explaining the summary of each task simply.	20
EPSS-L3	The target is a user who is a Web expert. Or a user who used this site many timesAssisting a user only he makes a mistake, for example, when inputting a world in a target object with mistake.	30

Result in WebAttendant

In WebAttendant, the only works required to develop EPSS-A was to create rules for defining the functions of EPSS. The authoring tool was used for the rule description. The EPSS developer created rules by simple operations. He or she only needed to follow the following 4 operational steps using the authoring tool:
Step1) select the condition part of the rule from the drop-down lists,
Step2) click a target object in the target Web page,
Step3) select the type of intervention from a drop-down lists, and
Step4) fill in a message that you want to show with an intervention.

We calculated the amount of work that needed to be done to develop EPSS-A. Table 4.4 shows the total number of operational steps using WebAttendant.

Result in built-in EPSS

To develop EPSS-A in built-in EPSS, the EPSS developer had to create several modules for required EPSS functions, using JavaScript and Java languages and embed it in each HTML Web page. Table 4.4 shows the line of the program created in the experiment.

Comparison

To develop EPSS-A using WebAttendant, the developer's only work was making rules by using the WebAttendant authoring tool. The total number of operational steps involved to make 43 rules was 103, and no special programming skill was required. In contrast, using a built-in EPSS framework, the developer had to create several EPSS modules using a programming language. We found that the total size of programs the developer had to modify were 2280 programming lines. Though it is obvious that the amount of work is lower in WebAttendant than in built-in EPSS because WebAttendant has already provided the basic modules, we think it is easier to develop an EPSS using WebAttendant than using built-in EPSS from the viewpoint of the user's mental workload for the knowledge of the programming languange.

4.4.3 Evaluation on reusability for different Web sites

Development of EPSS for different Web sites

We developed three different EPSS for three different Web sites once using the WebAttendant framework and another time using a built-in EPSS framework. The three different EPSS were the EPSS for Web training site (EPSS-A), which we already described in Section 4.4.2, the EPSS for US Census Bureau Web site (EPSS-B) as shown in Figure 4.4 and the EPSS for Web banking site (EPSS-C). We then compared the amount of work needed to complete for development of these EPSS when we used the WebAttendant framework compared to when we used a built-in EPSS framework. Table 4.3 shows the scenarios of EPSS services for each Web site and the number of functions (rules) created for each Web site. We consider reusability of rules individually as a condition part and an execution part. Table 4.5 shows the number of times reusing a module.

Result in WebAttendant

The only work needed to be completed to make the EPSS modules reusable was to change several parts of each rule that WebAttendant was going to reuse. We calculated the amount of work involved in modifying the EPSS-A to be reusable as EPSS-B or EPSS-C. The calculation was based on the total number of operational steps involved in modifying the rules.

Reusable rules that require no modifications are condition part independent of the Web site contents. The example of a reusable rule is a rule that defines the function

of "many-mouse operations (handling a user moving a mouse many times)". There are some reusable rules that we required some changes, which are dependent of the Web site contents. For example, the condition part of "click-hesitation" rule needs a target object for an instruction in a rule had to be changed. As show in Table 4.5, the total number of operational steps involved in modifying the reusable rules was 170 operational steps.

Result in built-in EPSS

We developed EPSS-A, EPSS-B and EPSS-C using a built-in EPSS framework. To reuse these EPSS, the developer had to modify the modules for each function using JavaScript and Java languages, and had to embed them in each module in each HTML Web page. We estimated the amount of work involved in EPSS development based on the total number of programming lines the developer had to modify to reuse these EPSS modules. As show in Table 4.5, the total programming lines modified for development of EPSS-B and EPSS-C were 422 lines.

Comparison

The work involved in making the EPSS reusable for other Web sites was 170 operational steps in WebAttendant and 442 program lines in Built-in EPSS. Moreover there is a difference in user's mental workload between the operation on the GUI authoring tool and the program modification on the editor. From the results of the experiment, we concluded that the WebAttendant framework is superior to a built-in EPSS framework in terms of its reusability for different Web sites.

4.4.4 Evaluation on reusability for different skill levels

Development of EPSS for different skill levels

We developed three different EPSS once using the WebAttendant framework and another time using a built-in EPSS framework. The three different EPSS were intended for users of three different skill levels. We developed the EPSS to "highly skilled users" as EPSS-L1, the EPSS to "intermediate-skilled users" as EPSS-L2 and the EPSS to "little-skilled users" as EPSS-L3. We compared the tasks involved in reusing the modules of these EPSS when we used a WebAttendant framework compared to when we used a built-in EPSS framework. Table 4.3 shows the scenarios of EPSS services for users with different skill levels and the number of functions (rules) created

for each Web site. We consider reusability of rules individually as a condition part and an execution part. Table 4.5 shows the number of times reusing a module.

Result in WebAttendant

The only works needed to complete to make EPSS modules to be reusable was to change several parts of each rule that WebAttendant was going to reuse. We calculated the amount of work involved in modifying EPSS-L1 to be reused as EPSS-L2 or EPSS-L3. The calculation was based on the total number of operational steps involved in modifying rules. Table 4.5 shows the results of reusability experiment in WebAttendant. As shown in Table 4.5, the total number of operational steps involved in modifying rules for reusing the modules was 85 steps. Reusable rules requiring no modifications are independent on the user's skill level. An example of such rules is "input-include-prohibited-char" that warns the user that there are some errors in the input area of the form. The errors in the form are crucial for any users. There are some reusable rules that required some changes which are dependent of the user's skill level. An example of such rules is changing the type of guidance in the execution part of the rule based on the user's skill level.

Result in built-in EPSS

To reuse the EPSS functions, the developer had to modify required modules using JavaScript and Java languages, and had to embed each module for each target object in each HTML Web page. We calculated the amount of work based on the total number of program lines we had to modify to make the EPSS modules reusable. Table 4.5 shows the results of reusability experiment in built-in EPSS. As shown in Table 4.5, the total number of programming lines that needed to be modified was 207.

Comparison

The work involved in making the EPSS reusable for other user-skill levels was 85 operational steps in WebAttendant and 207 program lines in Built-in EPSS. Moreover there is a difference in user's mental workload between the operation on the GUI authoring tool and the program modification on the editor. From the results of the experiment, we concluded that the WebAttendant framework is superior to a built-in EPSS framework in terms of its reusability for different skill levels.

4.4.5 Summary of the results

The summary of the results in the above-mentioned three experiments are as follows:

1. The EPSS developer's amount of work to develop an EPSS for the first time is lower in WebAttendant than in built-in EPSS.

2. The work involved in making the EPSS reusable for other Web sites is lower in WebAttendant than in built-in EPSS.

3. The work involved in making the EPSS reusable for other user skill levels is lower in WebAttendant than in built-in EPSS.

In the near future, EPSS for the Web-based applications used in the corporation will probably be outsourced like conventional business trainings. In that case, an EPSS framework with low development cost and high reusability for different Web sites or user skill levels would be effective for outsourcing service companies.

4.4.6 Future directions

Results of the experiments suggest the need for research on following areas:
(1) Management of rules

As the number of rules increases, keeping track of each rules and its operation condition become very complicated. Thus it is desirable to have a management tool to show the purpose and operating conditions of each rule.
(2) Dealing with dynamically generated pages

If the Web server is linked to a database and generates pages dynamically, management of access to objects in a Web page becomes complicated since the document structure may change. Even an Xpointer[12] will fail unless the original server defines the proper IDs. It is desirable to extend the rule-description technology to deal with some of the structural changes in such dynamically generated pages.

Table 4.4: The amount of development.

Type	Module name	Built-in EPSS Number of program lines	WebAttendant Number of operational steps
Event	load	12	1
	unload	32	1
	mouseover	34	2
	mouseout	45	2
	click	109	2
	long-focus	188	2
	long-no-keypress	217	2
	many-mouse-operations	193	2
	many-scroll-operations	170	1
	many-mouseover	159	2
	many-back-button-return	216	1
	click-hesitation	228	2
	input-include-prohibited-char	160	2
Guidance	pagechange	18	1
	window on	26	1
	balloon help on	132	1
	wizard	26	1
	web page on	258	1
	alert	4	1
TOTAL		2227	28

Table 4.5: Reusability.

Type	Function (rule) name	Experiment for different Web sites			Experiment for different skill levels		
		Reusing	Built-in	WebAt.	Reusing	Built-in	WebAt
Event	load	11	11	0	6	6	0
	unload	0	0	0	1	1	0
	mouseover	12	12	12	2	2	2
	mouseout	12	12	12	2	2	2
	click	4	4	4	0	0	0
	repreated-focus	2	12	2	0	0	0
	specified-focus-order	2	32	2	0	0	0
	specified-focus-unorder	3	48	3	0	0	0
	long-focus	0	0	0	1	4	1
	long-no-keypress	1	6	1	1	6	1
	many-mouse-operations	10	60	10	1	6	1
	many-scroll-operations	4	8	0	5	10	0
	many-mouseover	12	48	12	11	44	11
	many-back-button-return	6	24	0	7	28	7
	click-hesitation	9	36	9	6	24	6
	input-include-prohibited char	2	8	2	5	20	0
Guidance	page change	5	5	5	1	1	1
	window on	8	8	8	15	15	15
	balloon help on	49	49	49	13	13	13
	wizard	19	19	19	5	5	5
	web page on	14	14	14	5	5	5
	alert	6	6	6	5	5	5
TOTAL		191	422	170	92	207	85

Reusing: Number of times reusing a functions (rules).
Built-in: Number of modified program lines in Built-in EPSS.
WebAt.: Number of operational steps for modification in WebAttendant.

4.5 Conclusions

We proposed a framework for development of a content-independent EPSS called WebAttendant. In contrast to built-in EPSS, which is a part of Web content, WebAttendant's EPSS can be built independent of the Web content. We proposed the systen design and the system structure of WebAttendant. We have also evaluated its effectiveness. We conducted a series of experiments to test the effectiveness of WebAttendant. From the results of these experiments, we concluded that WebAttendant is highly effective as a platform of Web-based EPSS.

Preferences

[1] Stevens, G.H. and Stevens, E.F.: Designing Electronic Performance Support Tools: Talent Requirements, *Performance & Instruction*, Vol. 24, No. 2, pp. 9–11, (1995)

[2] http://www.w3.org/DOM/

[3] CoachWare, http://sterlingnet.com/sterling/coachware.htm

[4] Mmhelper, http://www.esmmi.com/product_more_weel.htm

[5] Microsoft Agent, http://msdn.microsoft.com/workshop/imedia/agent/

[6] QuickCards, http://www.epssinfosite.com/dd_qcard.html

[7] Duke-Moran, C., Swope, G, Morariu, J, and deKam, P.: Performance Support Case Studies from IBM, *International Society for Performance Improvement, Performance Improvement Journal*, Vol. 38, No. 7, (1999).

[8] American FactFinder, http://www.census.gov

[9] Furui, Y., Aoki, Y., and Hijikata, Y.: A Web proxy for embedding operation profiling and automatic navigation programs in Web pages, *Proc. of the 60th IPSJ 5S-8*, pp. 421–422, March 16, (2000).

[10] Aoki, Y., Ando, F., and Nakajima, A.: Web Operation Recorder and Player, *Proc. of The 7th International Conference on Parallel and Distributed Systems (ICPADS2000)*, pp. 501–508, (2000)

[11] Aoki, Y., and Nakajima, A.: User-Side Web Page Customization, *Proc. of the 8th International Conference on Human Computer Interaction (HCI International '99)*, Vol. 1, pp. 580–584, (1999).

[12] XML Pointer Language (Xpointer) Version 1.0 W3C Candidate Recommendation 7 June 2000, http://www.w3.org/TR/xptr

Chapter 5

Conclusions

This doctoral dissertation considered the technologies supporting the personalization in information browsing systems. There are four kinds of functions required for developing personalization in an information gathering system, which is the higher rank concept of an information browsing system: (1) user information gathering function, (2) user modeling function, (3) comparing and selecting function and (4) authoring function. Out of the four functions, this research studied on the user information gathering function and the authoring function.

For the user information gathering function, we developed a system called TextExtractor, which automatically extracts a text part that the user was interested in from the whole text of a page using the user's ordinary mouse operation. First, we surveyed the user's mouse operation performed during his/her usual Web browsing. In this survey, we found four kinds of operations related to the users' interests: text tracing, link pointing, link clicking and text selection. TextExtractor works for any Web sites because it is implemented as JavaScript and Java applet and inserted to HTML by the proxy server. First, TextExtractor acquires the DOM (Document Object Model) event and analyzes the sequence of the DOM events to detect the above-mentioned four kinds of operations. After that, it extracts the text part which is the target of the detected operations.

We conducted an experiment to see the effectiveness of TextExtractor. In this experiment, we saw if the extracted text by TextExtractor is actually the part the user was interested in. Five users participated in the experiment and browsed their favorite pages as usual. The result shows that the target text parts of every four kinds of operations include keywords the user was interested in at higher ratio (approximately 4%) than whole text of the page. We also compared TextExtractor with tf-idf which is the most popular keyword selection method. The result shows that TextExtractor

extracted the keyword that the user was interested in at about 1.4 times of accuracy. The result also showed that TextExtractor extract keywords at high accuracy even for pages with miscellaneous styles such as bulletin boards and link collections where tf-idf does not achieve its best performance. From these results, we concluded that the system can automatically acquire the part of the Web page the user was interested in from the user's ordinary mouse operation and its precision is better than that of other methods such as tf-idf. We expect that TextExtractor leads to the user's more frequent usage of relevance feedback and the accurate Web search.

For the authoring function, we considered the two kinds of features of the WWW: (i) it allows users to acquire information by moving from a page to another page by a link and (ii) it provides GUIs as a platform for building Web applications. We studied on the following two kinds of functions for the above features: (1) a verifying tool for user-navigation and (2) a framework for an EPSS on the Web.

The verifying tool for user-navigation checks the navigation rule which was described by the information provider. The most popular personalization method is a rule-based control. This needs the rule described by the human in advance. If there are some errors in the rule, this leads to the incorrect navigation. We developed a simple hypermedia system for testing the verifying tool and adopted the link hiding for the adaptation method. In this environment, we focused on a dead end and a loop problem. A dead end is the status, in which all links are hidden and the user cannot go anywhere after reaching a node with a navigation rule. A loop is the status, in which there is an unintended loop or the user cannot follow a loop that the information provider intended the users to follow. Generally loops are effectively used in the WWW. However the dynamic adaptation may hide some part of the loop or create unintended loops. This authoring tool automatically finds dead ends and loops by following the paths and investigating the rules.

We conducted an experiment to evaluate the authoring tool. This evaluation examined 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 and reduced the number of errors in the described navigation rules. In evaluation of the navigation error, an analysis of variance showed a significant difference at the 5% level of significance. There was no significant difference in the description time. However there was a relationship that as the description time becomes longer the error ratio gets smaller. Because the error ratio in describing rules with the authoring tool is smaller than that in describing rules without it, we also confirmed the authoring tool's effectiveness in description

time. Therefore we expect that the verifying tool for the user-navigation guarantees the correct navigation and helps the information provider in describing the navigation rules.

For a framework for an EPSS on the Web, we developed WebAttendant which allows the EPSS developer to develop an EPSS independent of the Web content. WebAttendant consists of a server and a client developed as JavaScript and Java applet program. Therefore it works for any Web sites. It also offers standard functions required for EPSS. The only work the EPSS developer has to do is describing guidance rule using the GUI authoring tool.

In order to verify the superiority of WebAttendant framework over a built-in EPSS in cost-efficiency of development, we ran two sets of experiments to evaluate two aspects of EPSS development. One experiment is for evaluating in an amount of work involved to develop an EPSS. We calculated the total operational steps involved in developing an EPSS when we used WebAttendant. Then, we examined the total size of the programs involved to develop the same EPSS when we used a built-in EPSS framework. Finally, We compared the total operational steps involved in EPSS development with WebAttendant framework to the total size of program involved in EPSS development with built-in EPSS framework. The result shows that it is obvious that the amount of work is lower in WebAttendant than in built-in EPSS. The other experiment is for evaluating in reusability. We developed several different EPSS for different Web sites and for users with different skill levels in a same Web site. We then calculated the amount of work when modules for EPSS functions to a specific Web site or a specific skill-level user were reused for other Web sites or other skill-level users as well. The work involved in making the EPSS reusable for other Web sites (skill-levels) was 170 (85) operational steps in WebAttendant and 442 (207) program lines in Built-in EPSS. We concluded that the WebAttendant framework is superior to a built-in EPSS framework in terms of its reusability. From these results, we concluded that WebAttendant is highly effective as a platform of Web-based EPSS.

In the future, more miscellaneous users with different backgrounds and skill-levels will participate in the WWW and more diversified applications will run on the WWW. We can even expect that not only on-line shopping services or reservation services but also the desktop application like a spreadsheet or word processor will work on the WWW. In such a situation, personalization will become still more important from a view of usability and business. We hope that the technology for acquiring the user information and the authoring tool proposed in our research will contribute to the growth of personalization.

Acknowledgements

This doctoral dissertation work was carried out at Department of Systems and Human Science, Graduate School of Engineering Science of Osaka University under the direction of Professor Shogo Nishida and at IBM Research, Tokyo Research Laboratory.

First I deeply thank my advisor Professor Shogo Nishida who have ingrained me a researcher's seed to invent ideas positively. I also thank him for giving me the opportunity to study in Nishida Laboratory and also for consulting me on personal related matters. I would like to express my gratitude to Professor Seiji Inokuchi and Professor Masahiko Yachida for serving as members of my thesis committee and for their invaluable comments on the thesis.

I express my thanks to Dr. Tetsuya Yoshida who eagerly discussed over my work. His suggestion made me more critical about my work. Thanks are due also to Dr. Naoki Saiwaki for giving me personal advises. I have good time with him in managing the laboratory like a network construction, a laboratory trip, and so on.

Acknowledgements must also be made to Mr. Amane Nakajima a program manager in IBM Research, Tokyo Research Laboratory. I was instilled the severity to work in my mind. This became the property which is irreplaceable for my research skill. I also thank to my coworkers in IBM Research, Mr. Yoshinori Aoki, Mr. Yohnosuke Furui, Dr. Toshio Sohya and Miss Yuko Ikehata for their great advices and their cooperation on my work. I would like to express my thanks to the staff in IBM Research who participated in the experiment of my work. I am happy to meet many wonderful coworkers of synchronous entrance into IBM. They are very kind to me for everything. They are so talented and have individuality that I was deeply encouraged from them. My work was influenced also from them.

I would like to extend my gratitude to Mr. Hachizo Yumoto, my grand father, Ms. Mariko Yumoto, my grand mother, and Ms. Kuniko Enomoto, my aunt for giving me the yell. I also want to be thankful to Miss Keiko Ishida. She is always there for me and makes me a bright feeling. I am also grateful to her for giving me the chance to get my PhD.

Lastly, I want to thank to my father and my mother for supporting me in every possible way until this time.

List of Publications

A. Journal Papers

1. Hijikata, Y., Yoshida, T. and Nishida, S.: Adaptive Hypermedia System for Supporting Information Providers to Direct Users through Hyperspace, *Transaction of IEEJ*, Vol. 120-C, No. 11, pp. 1720–1731 (2000). (in Japanese)

2. Komatsu, T., Hijikata, Y., Saiwaki, N. and Nishida, S.: Automatic Generation of Moving Crowd using Chaos and Electric Charge Model, *Transaction of IEEJ*, Vol. 121-C, No. 1, pp. 118–126 (2001). (in Japanese)

3. Watanabe, M., Yoshida, T. Saiwaki, N., Hijikata, Y. and Nishida, S.: An Image Based Support System for Web Page Design, *Journal of Human Interface Society*, Vol. 3, No. 4, pp. 73–83, (2001). (in Japanese)

4. Ikehata, Y., Souya, T. and Hijikata, Y.: Content-Independent EPSS with automatic context sensing on the Web, *Transaction of IPSJ*, Vol. 43, No. 2 (2002). (to appear)

5. Hijikata, Y., Aoki, Y., Furui, Y. and Nakajima, A.: Text Part Extraction based on Mouse Operation and Evaluation of Extracted Keywords, *Transaction of IPSJ*, Vol. 43, No. 2 (2002). (in Japanese) (to appear)

6. Hijikata, Y., Yoshida, T. and Nishida, S.: Supporting Information Providers for Rule-based Adaptive Hypermedia, *Interacting with Computers*. (submitted)

7. Wang, Y., Nozawa, H., Hijikata, Y., Nakatani, M. and Nishida, S.: Spatio-Temporal Data Management for Moving Objects, *Transaction of IEEJ*. (in Japanese) (submitted)

B. International Conference Papers

1. Hijikata, Y., Saiwaki, N., Tsujimoto, H. and Nishida, S.: A Dynamic Linkage Method for Hypermedia, *Proc. of 5th IEEE International Workshop on Robot and Human Communication (ROMAN'96)*, pp. 519–524 (1996).

2. Hijikata, Y., Saiwaki, N., Yoshida, T. and Nishida, S.: A Dynamic Linkage Method for Hypermedia and Its Design Support Tool, *Proc. of 7th International Conference on Human-Computer Interaction (HCI International '97)*, pp. 727–730 (1997).

3. Hijikata, Y., Takeuchi, H., Tsujimoto, H. and Nishida, S.: A Dynamic Linkage Method for Text Data Based on Self-Organizing Map, *Proc. of 6th IEEE International Workshop on Robot and Human Communication (ROMAN'97)*, pp. 420–425 (1997).

4. Hijikata, Y., Yoshida, T. and Nishida, S.: A Dynamic Linkage Method for Hypermedia and Its Design Support Tool, *Proc. of 1998 IEEE International Conference on Systems, Man, and Cybernetics, (IEEE SMC'98)*, pp. 1260–1265 (1998).

5. Hijikata, Y.: Estimating a User's Degree of Interest in a Page during Web Browsing, *Proc. of 1999 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC'99)*, No. IV, pp. 105–110 (1999)

6. Ikemoto, K., Hijikata, Y., Nakatani, M. and Nishida, S.: A Data Management Structure for Spatio-Temporal Walkthrough, *Proc. of Fifth International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies (KES'2001)*, pp. 175–180 (2001)

7. Hijikata, Y., Yoshida, T. and Nishida, S.: Adaptive Hypermedia System for Supporting Information Providers in Directing Users through Hyperspace, *Proc. of Third Workshop on Adaptive Hypertext and Hypermedia*, pp. 147–156 (2001).

8. Ikehata, Y., Sohya, T. and Hijikata, Y.: Content-Independent EPSS with automatic context sensing on the Web, *Proc. of The 2002 Symposium on Applications and the Internet (SAINT-2002)*. (to appear)

9. Wang, Y., Nozawa, H., Hijikata, Y., Nakatani, M. and Nishida, S.: Spatio-Temporal Data Management for Moving Objects, *Proc. of Pan-Yellow-Sea International Workshop on Information Technologies for Network Era*. (submitted)

C. Domestic Conference Papers (in Japanese)

1. Hijikata, Y., Saiwaki, N., Tsujimoto, H. and Nishida, S.: A Dynamic Linkage Method for Hypermedia, *Proc. of IEEJ Electronics, Information and Systems Conference*, pp. 601–606 (1996).

2. Hijikata, Y., Yoshida, T. and Nishida, S.: A Dynamic Linkage Method for Hypermedia Using Meta Data and User Model, *Proc. of the 57th Annual Conference of IPSJ*, No. 3, pp. 105–106 (1998).

3. Shinkai, D., Hijikata, Y., Yoshida, T. and Nishida, S.: A Method for Information Retrieval via User's Viewpoint, *Proc. of the 42th Annual Conference of Systems, Control and Information Engineers (ISCIE)*, pp. 589–590 (1998).

4. Hijikata, Y.: Estimating a User's Degree of Interest in a Page during Web Browsing, *Proc. of Annual Conference of JSSST*, pp. 349-352 (1999).

5. Furui, Y., Aoki, Y. and Hijikata, Y.: A Web Proxy for Embedding Operation Profiling and Automatic Navigation Programs in Web Pages, *Proc. of The 60th Conference of IPSJ*, pp. 421–422 (2000).

6. Furui, Y., Aoki, Y., Hijikata, Y., Souya, T. and Nakajima, A.: How to Apply the Technologies for Detection and Playback of Web Browser Operation to Marketing, *Proc. of IPSJ SIG-DPS*, 2000-DPS-97, pp. 421–422 (2000).

7. Hijikata, Y., Aoki, Y., Furui, Y. and Nakajima, A.: TextExtractor: Text Part Extraction Using Operation Logs on Web Browser, *Proc. of The 8th Workshop on Interactive Systems and Software (WISS 2000)*, pp. 201–206 (2000).

8. Wang, Y., Hijikata, Y., Nakatani, M., Nishida, S.: A Spatio-Temporal Data Management Structure for Identifying the Picture of Monitoring Camera, *Proc. of the 45th Annual Conference of the Institute of Systems, Control, and Information Engineers (ISCIE)*, pp. 285–286 (2001).

9. Wang, Y., Nozawa, H., Hijikata, Y., Nakatani, M. and Nishida, S.: Spatio-Temporal Data Management for Moving Objects, *Proc. of IEEJ Electronics, Information and Systems Conference*, Vol. II, pp. 605–608 (2001).

D. Lecture Notes

1. Hijikata, Y., Aoki, Y., Furui, Y. and Nakajima, A.: TextExtractor: Text Part Extraction Using Operation Logs on Web Browser, Interactive Systems and Software VIII, pp. 201-206, 2000. (in Japanese)

2. Hijikata, Y., Yoshida, T. and Nishida, S.: Adaptive Hypermedia System for Supporting Information Providers in Directing Users through Hyperspace, Springer Lecture Notes in Computer Science. (acceptted)

E. Patents

1. Hijikata, Y., Aoki, Y. and Nakajima, A.: TextExtractor, (1999).