The Computer Integrated Documentation Project: A Merge of Hypermedia and AI Techniques

Nathalie Mathé[1] & **Guy Boy**

NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035, USA
mathe@ptolemy.arc.nasa.gov

ONERA-EURISCO
Prologue1, BP 27-25
31312 Labege Cedex, France
boy@tls-cs.cert.fr

ABSTRACT

To generate intelligent indexing that allows context-sensitive information retrieval, a system must be able to acquire knowledge directly through interaction with users. In this paper, we present the architecture for CID (Computer Integrated Documentation), a system that enables integration of various technical documents in a hypertext framework and includes an intelligent browsing system that incorporates indexing in context. CID's knowledge-based indexing mechanism allows case-based knowledge acquisition by experimentation. It utilizes on-line user information requirements and suggestions either to reinforce current indexing in case of success or to generate new knowledge in case of failure. This allows CID's intelligent interface system to provide helpful responses, based on previous experience (user feedback). We describe CID's current capabilities and provide an overview of our plans for extending the system.

INTRODUCTION

Retrieving specific information from large amounts of documentation is not an easy task. It could be facilitated if information relevant in the current problem solving context could be supplied automatically to the user, in understandable terms and in a flexible manner (e.g., allowing the user to ask questions). This is a long-term goal of Computer Integrated Documentation (CID). As a first step towards this goal, we are developing an intelligent hypertext interface to help users browse through large documents in search of specific information [2].

CID has been developed on three aerospace applications: the Space Station Freedom Requirements documents, Space Shuttle mission control procedures manuals, and F-18 emergency procedures. These applications offer a wide variety of problems in technical documentation for both design and operations issues. This project is now three years old, and the current version of CID is used in a beta-test mode in several NASA centers. A typical screen of CID windows is presented in Figure 1. It is composed of a control panel that allows the user to control the entire library. Among its various capabilities, it also converts flat text documents into hypertext, indexes them, and records users' traces in the documentation. Both text and graphics capabilities are available within CID.

This paper provides a paradigm for information access based on the actual use of the hypertext system itself. First, it presents a new approach to incremental context acquisition in information retrieval that modifies existing relations between descriptors and referents by using on-line user feedback to either reinforce or correct the system's knowledge in case of success or failure. This feature allows the system to tailor itself to the user. Second, CID includes a mechanism that allows presentation of referents that are most likely to be useful, thus providing focus for the search in a hypertext database. (The user can, however, access all the descriptors and referents at any time). This model consequently improves the performance of the system. This paper is structured as follows. In the second section, the problem of browsing through large documents is reviewed. The

[1]Dr. Mathé is currently under contract 3004-47-2 to Recom Technologies, Inc.

third section presents the current CID system, and the fourth section describes how CID's subsystem IARC (Index Acquisition and Refinement according to Context) is able to acquire new knowledge. In the fifth section, some of the lessons learned from analyses of existing technical documentation systems, and from the design, implementation and the first tests of CID are given. Finally, the sixth section presents related work, and the seventh section discusses our system's limitations and our plans for future research issues.

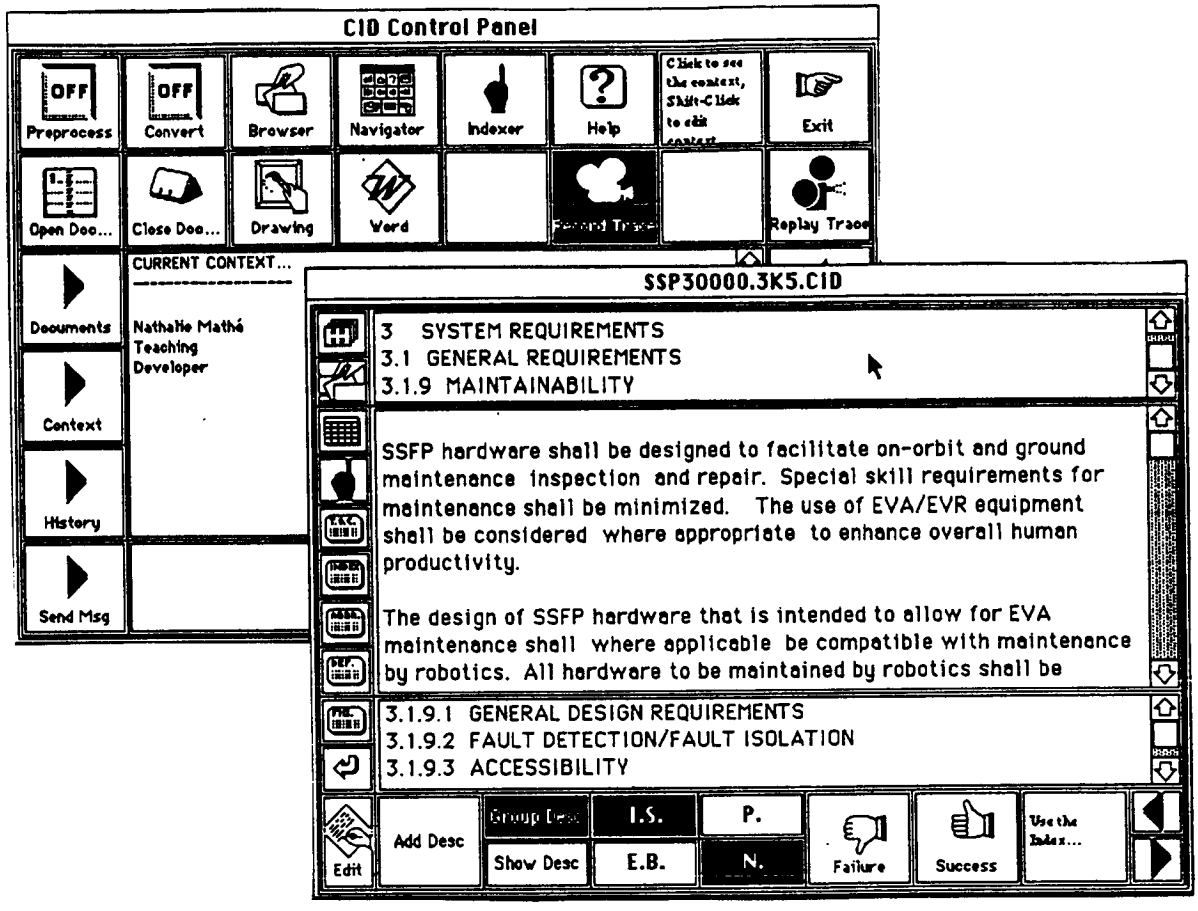

Figure 1. Part of a typical CID screen. In the lower-right window, the upper field includes the hierarchy of the displayed referent, the middle field is the actual text and the bottom field includes the next referents in the hierarchy. The upper-left window is the CID control panel.

THE PROBLEM

From analyses we have conducted during the design (and redesign) of CID, and from the current users' feedback, the following issues have become clear. When looking for specific information, people usually employ both the table of contents (hierarchical browsing) and the index of the available documents to guide their search. Even then, however, the search is not very focused, and it is common for users to examine several places in the documentation before finding the information needed. For example, when the index is used and a descriptor is looked at for the first time, the first page indicated is usually chosen. If the corresponding information turns out to be relevant for the user's purposes, the user is satisfied, and the retrieval process is finished. If, on the other hand, the corresponding information is not relevant, the user will normally go back to the index and try the second page proposed, and so on. This leads to a sequential trial-and-error decision process, which can be tedious and slow. To avoid repeating this lengthy process each time, people tend to build context around the descriptors already used. For instance, they tend to

remember that a particular referent was successful in a specific situation. This memory however does not generally last very long unless the same context occurs often.

CID has been designed to assist in this process by attempting to immediately provide the user with the information that is likely to be relevant to him/her *in context*. This paper presents an approach *to indexing in context*, in which indices are particular procedures that are modified incrementally *by experimentation*, remembering what the user found useful in a specific context. Using this approach, CID can provide tailored guidance to users browsing a document, by watching the users perform their browsing tasks. As Chen and Dhar [4] mention, to make large information banks more accessible by computer, it is best first to try and understand how reference librarians actually help users, and then to try to include these capabilities in on-line systems. The major problem in incorporating a model of user-librarian interactions into the system is the difficulty of acquiring the information for this model. CID's on-line knowledge acquisition approach allows semi-automatic acquisition of a librarian model.

Organization of Technical Documentations

In existing technical documentations, such as those common at NASA, information is structured hierarchically. Designing a complex system like the Space Station Freedom is an iterative process. Its documentation system is designed to handle a huge amount of information. It is organized around the Program Requirement Document (PRD), which establishes the highest level requirements associated with the Space Station Program. Generally, the other documents expand upon the topics expressed in the PRD. Each document includes the following major nodes: a description of the document, a preface, a table of contents, a body of text segmented into sections and subsections, an abbreviations table, a definitions table, and appendices. Each major node may contain a hierarchy of nodes. For instance, in the body of text, there is a hierarchy of sections. There are links between sections which are linear (section to next section) and non linear (reference to a section other than the next one). There are references to other major nodes within a document and sometimes to other documents. A *descriptor* [2] is any word, phrase, or piece of graphics which provides a meaningful "starting point" for a search in the documentation. A *referent* is the address of any part of text or graphics. Descriptors are organized hierarchically. In the current implementation, a referent is typically a document section. In this paper, a referent will also represent its content. In a regular book, the table of contents provides a list of descriptors (section titles). In this case, each descriptor corresponds to only one referent (a page number). The index also provides a list of descriptors, each of which usually has several referents, i.e., a sequence of page numbers. As a result, the user may need to look at several referents before finding the information needed.

Browsing through large texts

It takes years of training to be a flight controller in the Space Shuttle Mission Control Center. As part of this training, people learn to use a large corpus of documentation to solve problems. They develop a deep knowledge of the organization of these manuals in order to access the proper sections as quickly as possible. Currently the operational documentation used by flight controllers is paper-based. In the short term, the goal of CID is to help people access documentation on a computer more efficiently. One thing CID attempts to do is to help narrow the search through documents while allowing the full browsing freedom to which people are accustomed.

People typically use descriptors to retrieve information. As they start looking for information, they normally employ the *explicit* descriptors provided in the documents, e.g., table of contents or index. Unfortunately the descriptors provided are *context-free*, and, generally each descriptor can describe many referents, the problem of decidability introduces a major problem of backtracking that the user may not accept, especially when dealing with real-time operations. As a result, people

[2]The concept of a descriptor is very important in information retrieval [15]. Building such descriptors requires expertise in the domain of investigation. We are using a technique developed by Mark Zimmerman [20] that allows full-text extraction of words associated with their frequency in the text.

usually build implicit descriptors as they browse through documents, that is they build a cognitive representation of the documentation that provides relations between pieces of information and their approximate locations in the documentation. They remember that this particular piece of information was (or was not) very interesting in a special context. These cognitive maps are later used to guide their browsing task. They are thus context- and user-dependent.

Hypertext provides good support for browsing in documentation and for naturally building associative links between descriptors and referents. To automatically help the user, the problem thus becomes to "contextify" the links between documentation nodes, that is to provide relations between descriptors and referents that are valid in the current context. These relations will vary depending on the situation and the user. Providing a context for the referents reduces the number of possible referents for a descriptor that a user has to look at and thus narrows the search. In this paper, we present a technique to acquire context for these relations automatically. We define the context acquisition problem as reinforcement of current actions, as well as discovery of "abnormal conditions" and generation of recovery actions. Our system observes the user's actions during the browsing tasks, and, by noting whether a specific referent was considered a success or failure by the user in a particular context, it is able to refine the indices to reflect their context of use. In this way, our system automatically acquires the knowledge necessary to operationalize a user model: which indices are appropriate when, and for which user. This is a significant departure from current work in user modeling, where systems are able to obtain and refine user models [10, 12], but the way these models are exploited is *hardcoded* in the systems and thus inflexible [14].

AN ADAPTIVE DOCUMENTATION BROWSING SYSTEM

A browsing facility has been developed to help users search for specific information in the available documentation. CID, like the printed documentation, includes a table of contents and an index. When the user selects a descriptor, a menu of ordered referents pops up. These referents have been found successfully in the *same context* in past retrievals. The order of referents is based on the past success rate of each referent in this context. These referents can be very different among users and in various contexts.

CID has two major components: a hypertext system and a knowledge-based system. One of the major goals of this project was to keep documentation independent from the knowledge of how to use it. This latter knowledge is represented in the system in the form of *contextual links* containing preconditions (*triggering conditions* and *contextual conditions*) and a list of referents together with an indication of how often each referent was successful (a *reinforcement slot*), and a description of the situations in which the referent was *not* successful (called an *abnormal condition*).[3] An example of a contextual link is shown in Figure 2.

```
(TRIGG. COND.   (Descriptor-1))
(CONTEXT        (C1 C2 C3))
(ACTIONS        (R1 +5 ((AC1 1) (AC2 3)))
                (R2 +3)
                (R3 +2)
                (R4 +1))
```

Figure 2. Example of a contextual link. Descriptor-1 is valid if the context conditions (C1, C2, C3) are satisfied. The referent R1 has a *reinforcement slot* of +5, indicating that it has been successful 5 times and has been not found useful in two abnormal conditions AC1 -- with its

[3]The reinforcement slot and the abnormal condition can be seen as an indication of whether or not the user's *goal* for finding specific information was achieved.

reinforcement slot of 1 -- and AC2 -- with its reinforcement slot of 3. The referent R4 has been successful once.

Typically, in CID, a triggering condition is characterized by a descriptor. The selection by the user of a descriptor thus triggers a list of actions to be considered. When there are several descriptors in a contextual link, these descriptors are aliases or synonyms. The contextual conditions indicate under which conditions the actions presented in the contextual link are actually appropriate. Contextual conditions characterize the environment in which the retrieval has been made successfully. For instance, let us assume that one wants to retrieve some very specific information on the air conditioning in the main cabin of the Space Station. The first thing one may try is to browse the documentation with the descriptor "air conditioning." If the retrieval context can be specified, e.g., "you are a designer, you are interested in the connection of the air conditioning system, and you have very little information about the electrical circuitry in the cabin," then a more efficient search can be accomplished. The search will not be the same under another context, e.g., "you are an astronaut, you are in the Space Station, and you are freezing." Importantly, in our system, contextual conditions for a particular contextual link are learned *through experimentation.* Contextual conditions allow clustering of contextual links, and, when the system is operational, pruning of inappropriate contextual links.

The current context is set up at the beginning of the session. It can be a default profile attached to the user name and automatically set by the system after the login. It can be changed at any time by the user, or modified by the system following changes of sensor values if the documentation system is connected to a real-time system (e.g., in the case of documentation used in process control). In a given context, the user generally selects a descriptor to get a list of potential referents. When the user selects a referent R from this list, the system automatically activates the link between the selected descriptor and the referent R. This activation leads to the presentation of the referent R.

CID has two modes of operation, which correspond to the two modes of activity in documentation use: (a) *experimental browsing*: a casual approach, often seen in activities such as exploratory learning, in which the computer can take an active role by suggesting interesting information to be examined; (b) *intentional search*: a deliberate search for information to fill a particular need, e.g., to prepare a report or answer a specific question. Experimental browsing allows augmentation of the initial set of links between descriptors and referents.[4] Intentional search is used to refine existing contextual links of knowledge, by acquiring more contextual conditions or refining the existing ones. In the next section, we focus on this mode.

ACQUIRING INDICES BY EXPERIMENTATION

Following the above example, with the descriptor Descriptor-1 (triggering precondition) "air conditioning" and the current context (C1, C2, C3) "you are a designer, you are interested in the connection of the air conditioning system, and you have very little information about the electrical circuitry in the cabin." A contextual link is first triggered by the descriptor. As there are four possible referents in the documentation, the four referents, (R1, R2, R3, R4), are presented: "a list of the vendors of air conditioning systems," "a description of the air conditioning system," "a checklist of what to do when the air conditioning fails," and "a diagram of the electrical circuitry in the main cabin of the Space Station." The first one is not satisfactory in the current context: it is a failure. This is indicated by a mouse click from the user. The second and the third are also failures. Failure cases will be presented in the third paragraph of this section. Fortunately, the fourth one will give the information that is needed: it is a success. The system thus learns that R4 was

[4]Initial links can be built automatically assuming that descriptors are explicitly included in referents. Our system scans the hypertext database and extracts each descriptor together with a list of referents that corresponds to all the locations where this descriptor has been found. The corresponding contextual links generated this way are context-free. However, this automatic approach is generally not sufficient because some referents cannot be implicitly described by a descriptor included in the text. In this case, human intervention is necessary, i.e., the user can generate his/her own descriptors associated to particular referents.

successful in this context. At the next retrieval under the same context (C1, C2, C3), R4 will be presented automatically from Descriptor-1.

To observe the user, inputs to CID include user judgments on the success of actual retrievals. After a referent has been found, the user can select either "success" or "failure." The system automatically records this selection by adding +1 or -1 to the reinforcement slot attached to the original contextual link referent inferred by the descriptor used.

Suppose now that, using the same contextual link, a particular situation is observed in which the user indicates R1 as a failure. It would now be inappropriate to repeat this experience again and again. Instead, CID notes that an *abnormal condition* has been encountered, and this knowledge is added to the contextual link. Like actions, reinforcement values are also associated with abnormal conditions. Abnormal conditions can be seen as exceptions to the "normal" use of the contextual link. When a failure occurs, the system attempts to obtain from the user the reason for this failure. A list of previously acquired abnormal conditions is automatically presented to the user when he/she indicates his/her willingness to provide an explanation. The user may select one of the explanations provided or generate a new one. The selection is then processed automatically and kept in the corresponding contextual link as an abnormal condition.

If the abnormal condition is observed many times, its negation will automatically be added into the appropriate context for the contextual link, as the situation which used to be considered "abnormal" can be considered as "normal." A rote learning algorithm used to augment the system's knowledge has been presented in [1].

LESSONS LEARNED AND TESTS IN PROGRESS

We have analyzed [3] various uses of documentation systems available at NASA including Space Station Freedom (SSF) program requirements documents, and Space Shuttle operations procedures manuals. The former type of documentation has been called design documentation, and the latter operational documentation.

Design documentation is generally handled using keyword search. People find this very difficult in practice because keywords are used in a full-text search mode. Consequently, people using such systems come up with either hundreds of references or nothing, according to the recall/precision criterion [15]. We have found that CID's approach introducing the experimental browsing mode, allows the user to index referents with concepts that are not necessarily words or term-phrases included in the text. Another aspect is that current systems used at NASA have poor navigation capabilities (often none at all). People tend to construct their own cognitive maps of the documentation even if nothing is provided to make explicit the documentation topography. We have found that explicit maps of the documentation are very useful. These maps can be local ("where to go next?"), or global ("where am I?"). They can also present either the hierarchical structure of the documentation (local or global tables of contents), or the conceptual relationships between referents via descriptors (local or global conceptual indexes). An example of use of a contextual link as a local aid is provided in Figure 3.

Operational documentation systems (usually paper-based) are generally handled using tables of contents. Furthermore, expert users tend to develop robust search strategies based on experience and context. We have observed that, unlike the design documentation users, operational documentation users have very integrated cognitive representations of the documentation they use, i.e., they know its hierarchical structure and an extensive set of conceptual links. The reason is that operations people, in a Mission Control room for instance, are highly trained to solve problems using operational manuals. Furthermore, they update such documentation from their own experience (not only its indexing but also its content). CID will facilitate this painful and expensive process.

We are currently testing CID with SSF and Space Shuttle specialists in order to get more information on the level of acceptance of CID learning capabilities and behaviors by both design and operations people. Initial results indicates that CID is very useful both for indexing documents

in context, and for improving the revision process of the documentation itself. Context-sensitive information retrieval gives extended possibilities such as providing search expertise from other users, e.g., "what would John Smith do in this situation?"

The current version of CID is implemented on a Macintosh II Cx with HyperCard (version 2.1) and uses external functions in C. HyperCard is a good prototyping tool that is easily disseminated at NASA because of its widespread availability. This allows CID to be tested on a very large scale. Although we do not have quantitative results yet, our experience with CID to-date is very encouraging. The Space Station Freedom documentation application currently contains several Mb of text and hundreds of descriptors linked to the documentation.

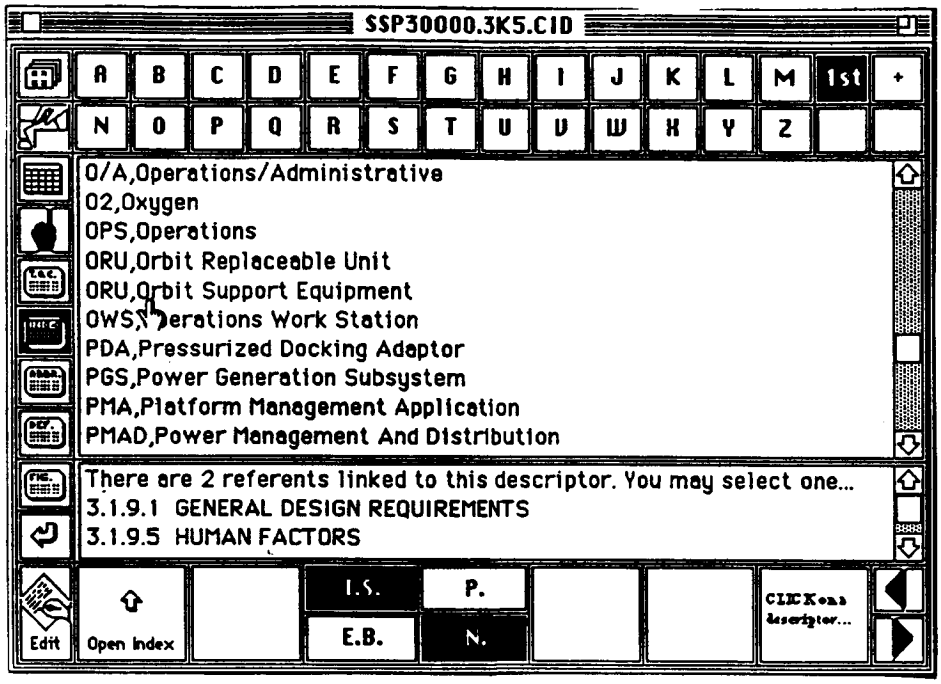

Figure 3. Use of an index in context. Here, the context was characterized by the type of user and the type of task. The user clicked on the descriptor "Orbital Replacement Unit". CID suggested two referents in the current context.

RELATION TO OTHER WORK

Semantic indexing has been investigated by several authors. Dumais et al. [7] propose a method for organizing nodes into a semantic structure on the basis of the overlap of the descriptors used in the referents. Stotts and Furuta [16] proposed a model of hypertext based on Petri nets. Their system enforces browsing restrictions, e.g., deactivates some links. Like CID, a medical handbook system described by Frisse and Cousins [8] separates the "index space" from the "document space." They have shown how some index architectures can be exploited for enhanced information retrieval, query refinement, and automated reasoning. Their index space model is based on inference using belief networks of descriptors. The I^{3}R system [5] uses a Bayesian inference network to acquire information about user's needs and domain knowledge. Crouch et al. [6] have used cluster hierarchies to help navigate in hypertext structure. All these contributions to information retrieval developed methods for refining links between descriptors and referents. However, the concept of context has not been presented explicitly in any of them.

Weyer [17] advocates the fact that information should be adaptable to the learner's preferences, and links should depend on the user's previous actions and current goals. This point of view supports our knowledge-based approach to hypertext. Other approaches have been developed to acquire and refine links on-line. For instance, Kibby and Mayes [11], in their StrathTutor hypertext system try to eliminate the need for exclusively manual methods for creating links between hypertext nodes by generating links based on knowledge acquired when the user browses through the system. Also, Monk presents a method for constructing a personal browser [13]. In this approach, the system monitors the user's navigation behavior and interrupts the user to ask whether it should add a node to the browser when it has been accessed frequently.

CID uses the concepts of context and abnormal conditions to learn from users. The work done on exceptions by Winston [19] and Williamson [18] is similar to our use of abnormal conditions. We have extended this approach with the use of dynamic reinforcement, and have incorporated these theoretical considerations in an actual implementation.

Finally, this work is a departure from current work on user modeling research, in which user models (possibly acquired automatically) are exploited by the system in predefined ways. As a result, these systems cannot update their user models based on experience. In contrast, our system can perform this updating automatically. In other words, our systems learns to *operationalize* user models automatically, based on experience.

CONCLUSION AND FUTURE DIRECTIONS

Current results have shown that hypertext is a good programming tool for the development of documentation systems. While hypertext systems increase accessibility, they do not provide any built-in selectivity mechanism. In other words, while non-linear or hypertext systems may dramatically increase the accessibility of information, this increased accessibility may magnify an already severe problem of selection [9]. For these reasons, our knowledge-based system technology can be very helpful in alleviating the selection problem and cognitive overhead of the user. Our approach to retrieval is unique because the design of contextual links to retrieve information is based not only on the way the documentation has been built, but also on user's information requirements and suggestions when they are operating systems. Thus, the user continually augments and refines the intelligence of the retrieval system.

Besides providing an intelligent interface for browsing large documents, the ability of our system to automatically acquire the context in which strategies are appropriate is significant. First, it allows the system to provide a *tailorable* browsing facility. Indeed, the system will learn which referents are to be presented for which user. Second, it shows that it is feasible to immediately incorporate the user's feedback into the system's knowledge, with the possibility of improving the system's performance. In this way, there is no need to collect and analyze large amounts of information about how users interact with the system because the system performs this task itself.

Many issues remain to be addressed. We briefly describe some of these here. First, formalization of the contextual conditions is still problematic. Contextual conditions should be minimal to avoid excessive calculations, but they must include as much information as possible to characterize the current situation. To solve this problem, future work will include the development of a context clustering mechanism. Second, we are extending CID to incorporate dynamic context, deduced from user's actions or associated to a dynamic environment. Third, we are developing a semantic similarity measure between referents, to let the user access information from a set of descriptors. (rather than a single descriptor). Finally, we are developing and evaluating a graphical contextual links browser to help navigate in large CID documents.

ACKNOWLEDGMENTS

I would like to thanks Celeste Plaisance, Mark Gersh and Kerry Soileau for their active support in evaluating CID with users at NASA HQ and at Mission Control, JSC. Thanks to Bharathi Rhagavan and Josh Rabinowitz for their contribution to CID implementation.

REFERENCES

[1] Boy, G.A., "Acquiring and refining indices according to context," Proceedings of the Fifth AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, November 1990.

[2] Boy, G.A., "Computer Integrated Documentation," MIT Conference on The Social Creation of Knowledge: Multimedia and Information Technologies in the University, held at MIT, April 6, 1991.

[3] Boy, G.A., "Computer Integrated Documentation," Technical Memorandum 103870, NASA Ames Research Center, Moffett Field, CA, September, 1991.

[4] Chen, H. and Dhar, V., "Reducing indeterminism in consultation: A cognitive model of user/librarian interactions," Proceedings of the Sixth National Conference on Artificial Intelligence, Seattle, Washington, July 1987.

[5] Croft, W.B. and Turtle, H., "A Retrieval Model for Incorporating Hypertext Links," Hypertext'89 Proceedings, pp. 213-224, ACM press, New York, 1989.

[6] Crouch, D.B., Crouch, C.J. & Andreas, G., "The Use of Cluster Hierarchies in Hypertext Information Retrieval," Hypertext'89 Proceedings, pp. 225-237, ACM press, New York, 1989.

[7] Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwester, S. and Harshman, R., "Using Latent Semantic Indexing to Improve Access to Textual Information," Proceedings of the ACM CHI'88, pp. 281-285, Washington D.C., May 15-19, 1988.

[8] Frisse, M.E. and Cousins, S.E., "Information Retrieval from Hypertext: Update on the Dynamic Medical Handbook Project," Hypertext'89 Proceedings, pp. 199-212, ACM press, New York, 1989.

[9] Jones, W.P., "How do we distinguish the hyper from the hype in non-linear text ?" Proceedings of INTERACT'87, Elsevier Science Publisher, Holland, 1987.

[10] Kass, R. and Finin, T., "Rules for the implicit acquisition of knowledge about the user," Proceedings of the Sixth National Conference on Artificial Intelligence, pp. 295-300, Seattle, Washington, July 1987.

[11] Kibby, M.R. & Mayes, J.T., "Towards Intelligent Hypertext," Hypertext Theory into Practice, McAleese, R. (Ed.), Albex, 1989, pp. 164-172.

[12] Kobsa, A., "A taxonomy of beliefs and goals for user models in dialog systems," In User Models in Dialog Systems, Kobsa, A. & Wahlster,W. (Eds.), Springer Verlag, Symbolic Computation Series, Berlin Heidelberg New York Tokyo, 1989.

[13] Monk, A., "The personal browser: A tool for directed navigation in hypertext systems," Interacting with Computers, 1, 2, 1989, pp. 190-196.

[14] Paris, C.L., "The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise," PhD thesis, Columbia University Department of Computer Science, 1987.

[15] Salton, G., "Automatic Text Processing: the transformation analysis, and retrieval of information by computers," Addison Wesley, Redding, Ma, 1989.

[16] Stotts, P.D. and Futura, R., "Adding Browsing Semantics to the Hypertext Model," Proceedings of the ACM Conference on Document Processing Systems, pp. 43-50, Santa Fe, NM, December 5-9, 1988.

[17] Weyer, S.A., "As we May Learn," In Interactive Multimedia: Visions of Multimedia for Developers, Educators, & Information Providers, Aubron, S. & Hooper, K. (Eds.), Microsoft Press, 1988, pp. 87-103.

[18] Williamson, K.E., "Learning from exceptions in databases," Machine Learning: A Guide to Current Research, Mitchell, T.M., Carbonell, J.G. & Michalski, R.S. (Eds.), Kluwer Academic Publishers, 1986.

[19] Winston, P.H., "Learning by augmenting rules and accumulating sensors," Proceedings of the International Machine Learning Workshop, pp. 22-24, Monticello, Illinois, June 1983.

[20] Zimmerman, M., "TEX version 0.5," Technical Report, Silver Spring, MD., 1988.