Query-free Information Retrieval

Peter E. Hart and Jamey Graham
RICOH California Research Center
2882 Sand Hill Road
Menlo Park, CA 94025
Email: fixit@crc.ricoh.com

Abstract

We introduce query-free information retrieval, a paradigm in which queries are constructed autonomously and information relevant to a user is offered without explicit request. Query-free methods offer an apparently new approach for integrating knowledge-based applications with legacy databases. We describe a fielded system, FIXIT, which integrates an expert diagnostic system with a pre-existing full-text database of maintenance manuals. The reported results suggest that query-free information retrieval can liberate the user from burdensome information retrieval activities while incurring only modest system development costs and minimal run-time costs.

Introduction

A fundamental real-world challenge for designers of cooperative information systems is to provide a useful integration of computational and data resources while at the same time satisfying several apparently conflicting objectives. While some objectives, like minimizing life-cycle costs or run-time costs, are typical of most system development activities, cooperative information systems introduce additional challenges: The integration of several large system components usually increases the capabilities available to end users compared with those provided by any single component. But how will end users access these capabilities? Will it impose additional burdens on users to invoke or control them? Will users be distracted from their "primary" or "base" application task by the features and functions of "secondary" or "supporting" system components?

Expert systems designers in particular face these problems because modern expert systems are seldom fielded as isolated, independent software applications. Instead, they are most frequently embedded in some larger environment that presents both opportunities and challenges for sub-system integration.

This paper reports work aimed at addressing some of these questions in a specific system context. We describe a fielded system, FIXIT, that integrates a probabilistic expert system with a full-text database of maintenance documentation. This legacy database is the online source of the actual printed manuals used by technical support personnel. In FIXIT, it entirely replaces the "help-text" found in conventional expert systems with far more comprehensive and up-to-date reference material.

FIXIT users are either service technicians or customer support representatives. With appropriate knowledge bases, they might be either troubleshooting a complex copier or helping a customer solve a less-complicated problem over the phone. In either case, because the expert system directly supports the user's diagnostic goal, we regard it as the base application.

The dialog between user and base application establishes a context and from time to time generates a need for additional diagnostic and repair information. In principal, such information could be made available directly from the on-line database, and a motivated user might

wish to access the database directly and search for relevant topics. Of course, this requires familiarity with access paths and query procedures and also might be a distraction from the primary task. Our challenge is to provide the user with the benefits of a combined expert and full-text database system without requiring knowledge of query procedures, without distracting attention from the diagnostic task, and without imposing noticeable runtime costs.

We have approached this challenge by introducing an intelligent agent that analyzes interactions between user and expert system and automatically constructs database queries based on this analysis. The user is unobtrusively notified when information relevant to the current diagnostic context has been returned, and may immediately access it if desired. From the user's perspective all database machinery is entirely transparent; indeed no formal query language is even made available. Hence we term this approach query-free information retrieval.

As we hope will be apparent from what follows, the introduction of the intelligent agent additionally offers one solution to a fundamental problem facing designers of cooperative information systems: How can legacy systems of substantial complexity be integrated within a larger system context? By requiring that all interactions with the legacy database be mediated by the agent, we have been able to isolate the database system cleanly while still supporting query-free information retrieval.

FIXIT is comprised of the three subsystems already mentioned: the probabilistic expert system, the legacy full-text database system (to which we added a new, semantically-based, indexing structure that supports limited natural language queries), and the intelligent agent that effectively integrates them. The following sections describe these system components, provide implementation details, illustrate the runtime behavior of FIXIT, report on operational experience, and close with some observations about query-free information retrieval and the potential for generalizing the underlying paradigm.

FIXIT's System Components

We first describe the probabilistic expert sub-system and the information retrieval subsystem. Before briefly describing these, we stress that our purpose was not necessarily to advance the capabilities of the individual components or indeed even to exploit fully the best current technology; instead, we focus on their integration.

Expert System Component. For our purely diagnostic applications, we elected to use the belief net (or Bayesian net) representation that has received considerable attention in recent years[1].

Belief networks use conditional probabilities of the form $p(s_j|f_i)$ to represent associations between a fault f_i and an observable symptom s_j that it may produce. A knowledge base for a belief net expert system consists principally of a collection of conditional probabilities of this form*.

The relations among faults and symptoms are conveniently represented as a directed acyclic graph as shown in Figure 1. Nodes repre-

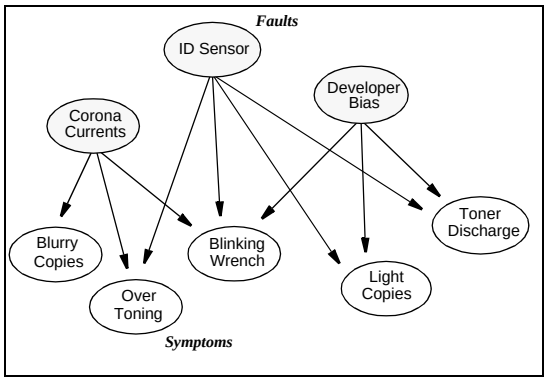

Figure 1. A simplified belief net for copier diagnosis

* We omit from this brief summary the usual discussion about conditional independence assumptions or the introduction of the higher order conditional probabilities often encountered in practice.

sent either faults, observable symptoms or unobservable internal variables. Arcs are informally "causal", pointing from underlying faults to the symptoms they may cause. Formally, arcs correspond exactly to those conditional probabilities required by the expert to represent the important probabilistic dependencies among faults, observables, and unobservable internal variables of the problem domain.

At knowledge engineering time the expert specifies the conditional probability of observing a particular symptom given the occurrence of some fault. At runtime these conditional probabilities are inverted: Given some sequence $\{s_1, s_2, \ldots, s_j\}$ of observed symptoms, we compute for each possible fault f_i the conditional probability $p(f_i|s_1, s_2, \ldots, s_j)$ of that fault given the sequence of observations. At each stage in the sequence a decision is taken whether to accept the currently most probable fault as the diagnosis or to make another observation. That decision can be based on either formal utility theory, informal heuristics, or the user's own judgment.

Information Retrieval Component. The information resources available in our setting are contained in a full-text database (including diagrams represented as bitmaps) of maintenance reference manuals, training manuals and frequently-modified maintenance notes and updates. Fortunately, this mass of material is very well structured by a chapter, section and subsection organization imposed by its authors. We refer to these organizational units collectively as topics. Authors take great care to choose the literal name of each topic to be as descriptive as is reasonably possible.

Our *information retrieval* (IR) subsystem[2] focuses on topics as the primitive elements to be returned to the user. Topics can be searched for, located and retrieved only by referencing the collection of their literal names. We limit ourselves to this level of representation out of respect for the difficulties of 'understanding' a full text-and-graphics database at a deeper level.

Operating as an isolated component, the information retrieval subsystem can in principal accept restricted natural language queries containing terms from a limited vocabulary. Queries are parsed to produce semantic constituents[3] that are used as the search terms for accessing the database.

Even though we have simplified matters by restricting attention to the level of database topics and by using a limited vocabulary, some additional machinery is still needed for two reasons. First, though topic names are descriptive they are frequently elliptical. For example, a topic named "Removal" can be understood only as a sub-topic of, say, "Fusing Unit". Second, a direct string match against literal names is too restrictive; a change in nomenclature might require burdensome system updates.

We deal with both these complications by appealing to our semantic representation in combination with a *Table of Contents*, or ToC, tree that captures the organization of the maintenance documentation (Figure 2). A ToC tree

```
3.  Fusing Unit
     3.1 Cleaning
     3.2 Removal

4.  Corona Wires
     4.1  Charge Corona Wire
          - Connector -
          - Adjustment -
     4.2  Transfer & Separation Corona Wires
     4.3  Prequenching Corona Wires
5.  ID Sensor
     5.1  Adjustment
     5.2  Removal
```

Figure 2: A fragment of the table of contents tree

data structure mirrors the form of the Table of Contents of the maintenance documentation, with each node in the tree corresponding to a topic in the documentation. However, the node contains not the literal topic name, but instead contains the constituents obtained by parsing the topic name.

Retrieval is performed by matching query

constituents top-down against ToC constituents. A partial match at any level of the ToC-- i.e., a match between at least one constituent of a query with at least one constituent of a ToC node-- triggers a recursive attempt to match additional query constituents lower in the ToC tree. Proceeding down any single path through the tree, we retrieve the topic corresponding to the deepest node at which a match is obtained. This matching procedure propagates information down the tree from higher to lower nodes and effectively handles ellipsis like that mentioned earlier. It returns multiple topics if matches are found on separate paths, but always returns only the most specific topic along any single path.

As an example of how the ToC IR system works, Figure 3 depicts the fault node Corona Currents Out of Adjustment and the results of matching it against a portion of the ToC tree. The IR system uses a matching function which produces a results vector for each comparison. Each value in the vector directly corresponds to a concept (at that position) in the semantic pattern. This value will reflect the amount of similarity between a semantic pattern concept and the constituents which make up the ToC topics. For simplicity, we use 0.0 to mean "No Match" and anything between 0.8 and 1.0 to mean a "Valid Match".

The figure displays two sets of results for each topic the semantic pattern is compared with: local and global. The local results represent the comparison between semantic pattern and the individual topic.

The global result vector depicts a component-wise merge of local and global results which effectively propagates the context of parent topics to subordinate topics throughout the ToC. The straight arrows drawn from the local to the global results show how the values are passed on to create what amounts to be the most contextually rich version of the results vector.

For example, given the semantic pattern in the figure, our first concept match comes when

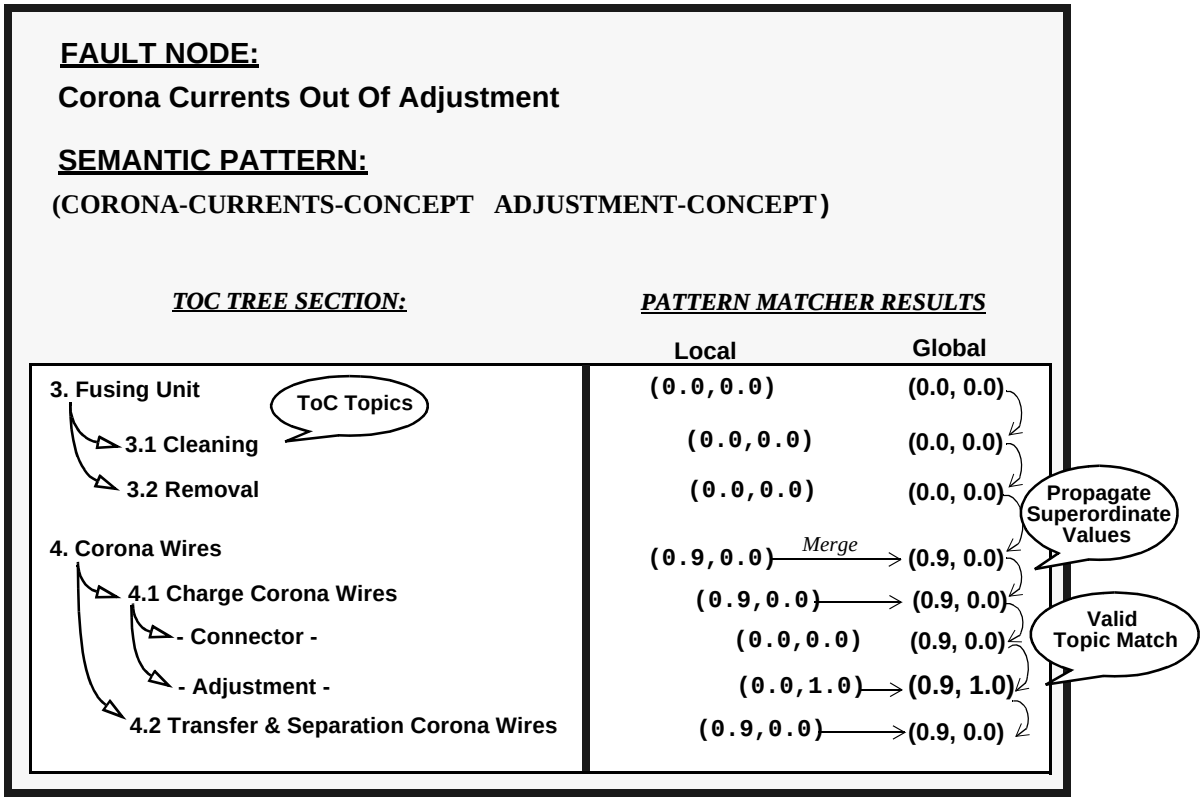

Figure 3 - ToC Information Retrieval Example

we compare the semantic pattern to the topic "4. Corona Wires". The resulting vector, (0.9, 0.0), represents a Valid Match between the first query constituent, CORONA-CURRENTS-CONCEPT, and No Match between the second query constituent ADJUSTMENT-CONCEPT (the pattern matcher produces a value of 0.9 for the above match because the concepts CORONA-CURRENTS and CORONA-WIRE are similar but not exact). Next, we recursively process the nodes subordinate to "4. Corona Wires", by propagating the parent vector, (0.9, 0.0), to the subordinate topics so that the full context of these topics can be realized when comparing them with the semantic pattern.

When the search compares the semantic pattern with the subtopic "- Adjustment -", we get a local match for the second constituent in the semantic pattern, (0.0,1.0). By merging the current local results with the current global results, (0.9, 1.0), we produce a Valid Match, utilizing the context provided by a superordinate topic.

We have informally assessed the precision and recall of the ToC information retrieval subsystem by running about 60 successfully-parsed queries against a database of almost 400 topics. We estimated the precision to be about 85% and the recall about 95%. While the sample size and rigor of our evaluation process might leave something to be desired, we certainly established that the subsystem is sufficiently robust to support the pursuit of our main experimental interest in query-free information retrieval.

Autonomous Query Construction. With the belief net and ToC subsystems as building blocks, we can frame our central issue: How does the context defined by the interactions between the user and the expert system provide clues about what documentation is currently most relevant? Put slightly differently: Can we guess what query a user would construct if he or she had a model of what information was available, understood the query language, and chose to type a request? If we can construct appropriate queries autonomously, our existing machinery is well able to do the rest of the job*.

Our approach to this problem relies on the twin concepts of *focus of attention* of the user and *support for a hypothesis.* An appeal to these concepts in turn depends on the fact that the technician invoked the expert system with a single, well-defined goal: identifying the cause of the copier malfunction.

Activated Faults. We assess the technician's focus of attention through the use of the fault probabilities computed by the expert system. We define an *activation predicate,* or AP, to focus attention on a manageably small number of faults from among possibly hundreds of diagnostic alternatives. It is easy to construct many AP definitions ranging from simple static thresholds to more elaborate comparisons that take recent history into account. For our initial experiments, a fault node in the belief net satisfies the AP simply when its current probability (i.e., its conditional probability given all symptoms observed thus far) is at least 0.03.

An activated fault node autonomously generates a request for further information in a natural way. The node name or descriptor is treated as a query, the ToC information retrieval machinery is invoked and a set of topics (possibly null) is returned. For this machinery to work successfully, the knowledge base designer and implementor must take a little care so that node names have some reasonable resemblance to the standard terminology employed in the maintenance documentation and the ToC. This is rarely difficult since knowledgeable people in a given field, whether they are documentation specialists or other experts, usually share a common vocabulary. Certainly, relying on even a loosely-controlled conven-

* A more sophisticated approach would also use interactions with the full-text database, not just with the expert system, to establish a context.

tional vocabulary is less problematic than dealing with uncontrolled queries from casual users.

Supporting Symptoms. With activated faults providing an operational definition of the user's focus of attention, we next want to operationalize a definition of their support. If the expert system were based on formal logic, the notion of support for a proof would be well-defined. In our probabilistic expert system we have less theoretical guidance and adopt a pragmatic approach.

We use the computational results of the belief net subsystem directly to obtain a measure of how strongly associated a symptom is with an activated fault. Suppose the first j-1 symptoms $\{s_1, s_2, \ldots, s_{j-1}\}$ have already been observed and we now observe the jth symptom s_j. For each activated fault f_i we easily compute Dp_{ij} as $p(f_i|s_1, s_2, \ldots, s_j) - p(f_i|s_1, s_2, \ldots, s_{j-1})$. In words, Δp_{ij} measures the marginal effect on f_i of observing s_j in the context of the previous j-1 observations.

As with the AP, it is easy to construct many alternative definitions of a *support predicate*, or SP, that focuses attention on significant values of Δp_{ij}. After some experimentation we settled on a simple static threshold, 0.01, applied to $|Dp_{ij}|$; i.e., we say a symptom s_j supports an activated fault f_i when $|Dp_{ij}| > 0.01$.

The support predicate leads directly to autonomous query construction just as the activation predicate does. The names or descriptors of nodes that support an activated fault are treated as queries, the ToC machinery is invoked, and a set of topics is returned.

Our definition of SP is obviously sensitive to the order in which symptoms are observed. A symptom s_j that strongly supports some fault diagnosis in one sequence of observations may be irrelevant in another sequence. This will occur whenever s_j is largely unnecessary or redundant given previous observations. From an information retrieval standpoint we consider this to be desirable behavior, arguing that the user is most likely to be interested in symptoms that substantially effect the diagnosis in the context of previous observations.

Connecting Faults and Symptoms. The combination of activation and support predicates enables us to take advantage of the opportunity -- not rare-- to identify database topics that connect faults and symptoms directly. This is simply accomplished by noting when a topic retrieved for an activated fault node is also present among the topics retrieved from a supporting symptom node-- i.e., by intersecting the sets of topics retrieved for each node individually. We call such co-occurring topics primary topics. We call a topic secondary if either (i) it was retrieved from an activated fault node, or (ii) it was retrieved from any symptom node asserted by the user.

To summarize, the foregoing concepts lead directly to the construction of an intelligent agent supporting query-free information retrieval. The agent monitors symptoms asserted by the user and employs the activation predicate to establish the user's focus of attention. The support for activated nodes is computed and queries based on asserted nodes, activated nodes, and supporting nodes are issued. Topics returned are identified as primary if the necessary set intersections are non-null, and are identified as secondary otherwise. Incidentally, the identification of a topic as primary or secondary affects only our user interface design; we alert the user to the availability of both types, but invite attention to primary topics when they are available.

FIXIT Implementation

FIXIT's component subsystems originated from different sources. For the belief net expert system shell we use DXpress™, a commercial product currently available from Knowledge

Industries*. Several probabilistic knowledge bases were built in collaboration with RICOH technical experts, and cover a variety of the most widely distributed RICOH products. (For our illustrative example in the next section we use the simplest fax machine knowledge base, which covers about a hundred or so malfunctions.) The full-text database is prepared and maintained by RICOH operational staff. The ToC information retrieval system was developed by one of us[2].

Fully-integrated versions of FIXIT run under several versions of Windows, including a Japanese version. The actual implementation differs from the foregoing slightly simplified description in only a couple of respects.

First, while the system would run well exactly as described, we can increase the robustness of information retrieval by augmenting the set of node descriptors beyond the minimum required for the belief net. The expert, at design time, simply specifies additional descriptors that might be semantically related to the node. This approach was adopted instead of any of several much more complex, but better-founded, alternatives.

Second, as the reader has probably already noticed, the universe of individual queries is limited to the node names (or node descriptors) in the belief net. This suggests parsing all possible queries at compile time, invoking the ToC retrieval machinery, and returning for each node a pointer to the relevant full-text database topic(s). Accordingly, substantially all the computation required to support information retrieval is performed at compile time; only the computationally trivial tasks of computing the AP, the SP, and intersecting small sets are performed at runtime.

An Illustration of Fixit Behavior. We illustrate FIXIT's run-time behavior with two brief, related examples.

Figure 4a shows a FIXIT window as it appears to a customer support representative during a diagnostic session. The top left quarter of the window displays a hierarchical menu containing all possible observable symptoms. Selecting first a symptom category (e.g. "Indicators") and then a specific symptom (e.g. "Clear Copy Indicator") results both in asserting that symptom as "observed" and posting it to the current history of the session (bottom left). The expert system computes the probability of each of the hundred or so faults given this symptom, and displays the leading candidates in the lower half of the window ("Possible Problems") along with their probabilities and fault categories. The agent indicates, via a text icon, that relevant documentation was found for some faults.

There are three kinds of icons used by the agent to indicate that relevant documentation is available for a fault. The primary text icon, as previously mentioned, indicates the availability of topics relevant to both the fault and to one or more of the observed symptoms. The secondary text icons represent the availability of documentation relevant to the fault. (We use two different icons to denote "many" or "few" available topics.)

In figure 4b we see that selecting the text icon results in a new window containing all topics found to be relevant to the fault "Paper Nonfeed-RX". By selecting a topic in the list we can see which document this topic comes from: "User's" manual.

Figure 4c shows that selecting a topic places the user in a documentation browser, which for convenience uses two windows to display the images and text separately so that they may be scaled and viewed independently.

In addition to serving as the interface to FIXIT's information retrieval component, the browser also provides access to the table of contents for this document, to all documents in the library for this particular machine, and to convenience features like bookmarks. In our example, the retrieved topic "1. Replacing Paper" is indeed relevant to the fault "Paper Non-

*http://www.kic.com/

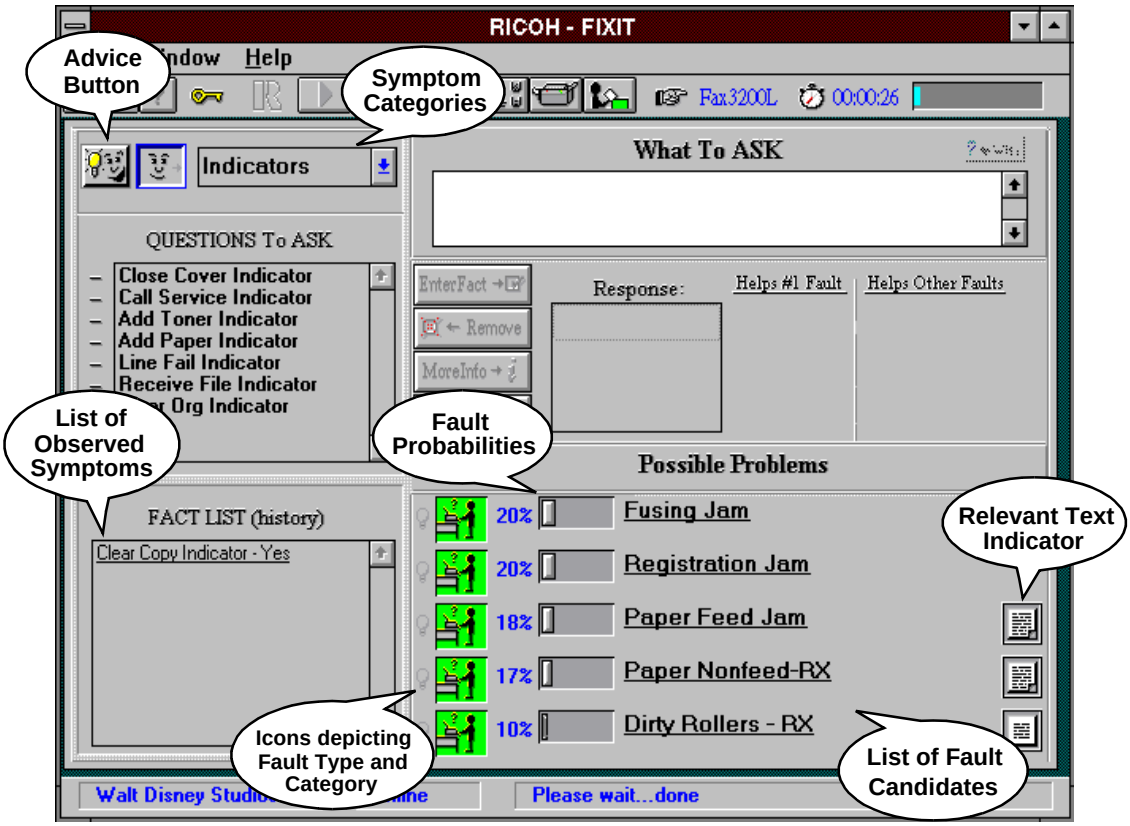

Figure 4a. FIXIT's Diagnostic Window

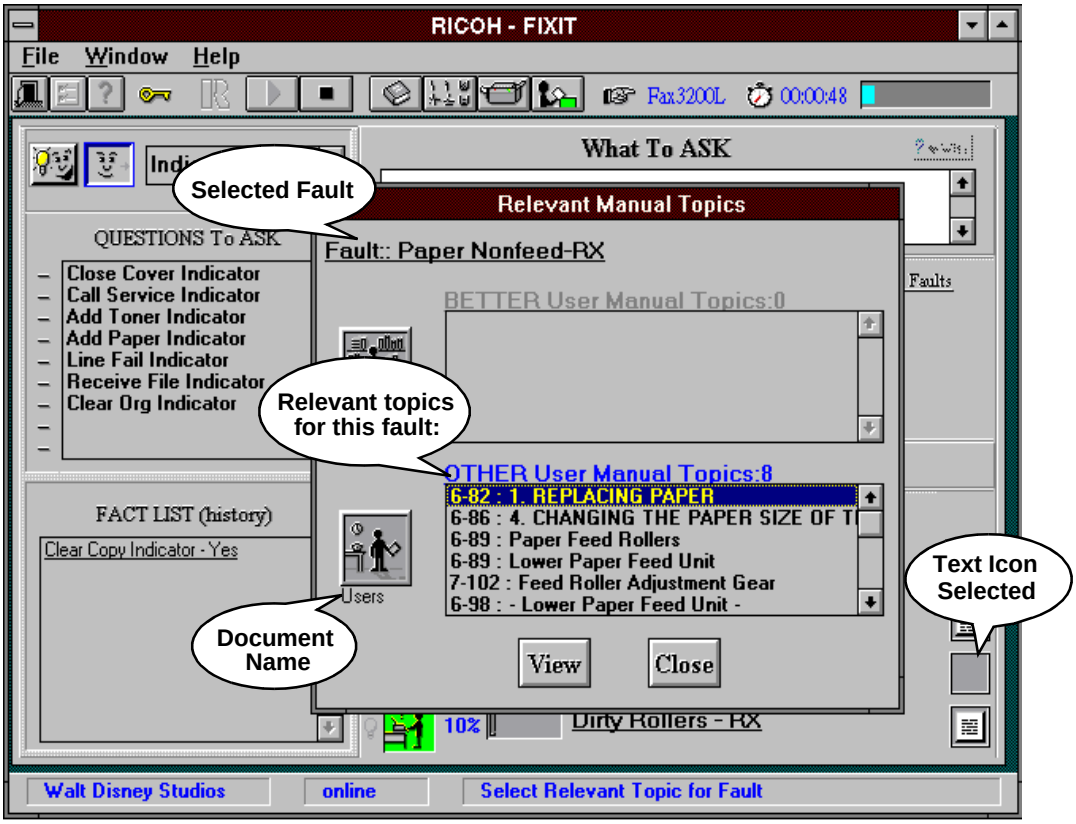

Figure 4b. Relevant topics for a fault

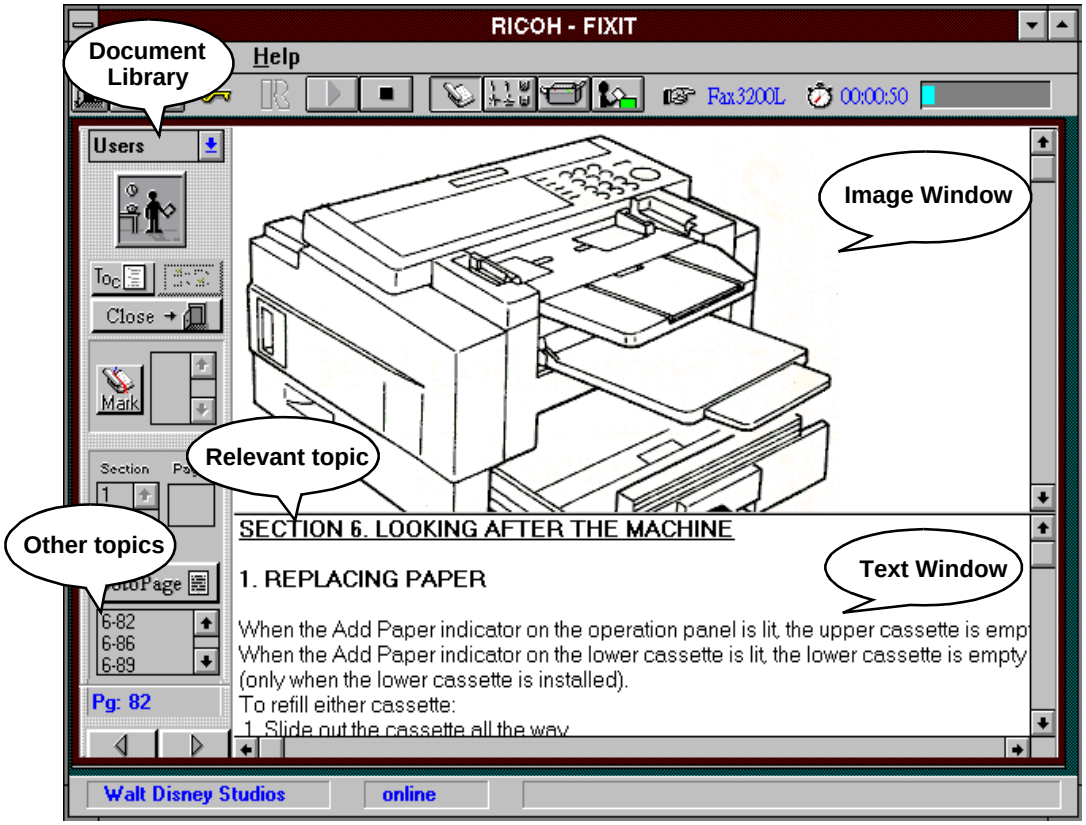

Figure 4c. Documentation Browser

feed-RX".

The context-sensitive nature of FIXIT's information retrieval can be illustrated by continuing the diagnosis. Figure 5a presents the same diagnostic session after asserting a few more symptoms. The number one fault is now "Paper Nonfeed-RX". However, the text icon for this fault has changed to indicate that there are primary topics available. Figure 5b shows the available topics and Figure 5c shows the topic retrieved, which is indeed relevant to both the "Paper Nonfeed-RX" fault and the "Chk Cassette Area-No" observed symptom.

Field Experience

In January of 1995 an alpha test was begun at a RICOH customer support center. Both the percentage of calls successfully resolved and the average time required to resolve them greatly surpassed the expectations of operational managers. Accordingly, in June 1995 the system was put into production use at one RICOH center.

In January 1996, following continuing effective use, the system was rolled out to additional RICOH locations. At the same time, knowledge engineering activities were expanded to increase the coverage of RICOH's broad product line, and development of a Japanese version was begun. As of this writing it seems fair to say that, viewed from the standpoint of operational managers (rather than from only our perspective as researchers and developers), the system is successful.

Discussion

We have introduced query-free information retrieval, an apparently new paradigm in which queries are constructed autonomously and information relevant to a user's task context is offered without explicit request. FIXIT, begun as a research project in cooperative information systems, has progressed from early concepts, to a research prototype, to a fielded commercial

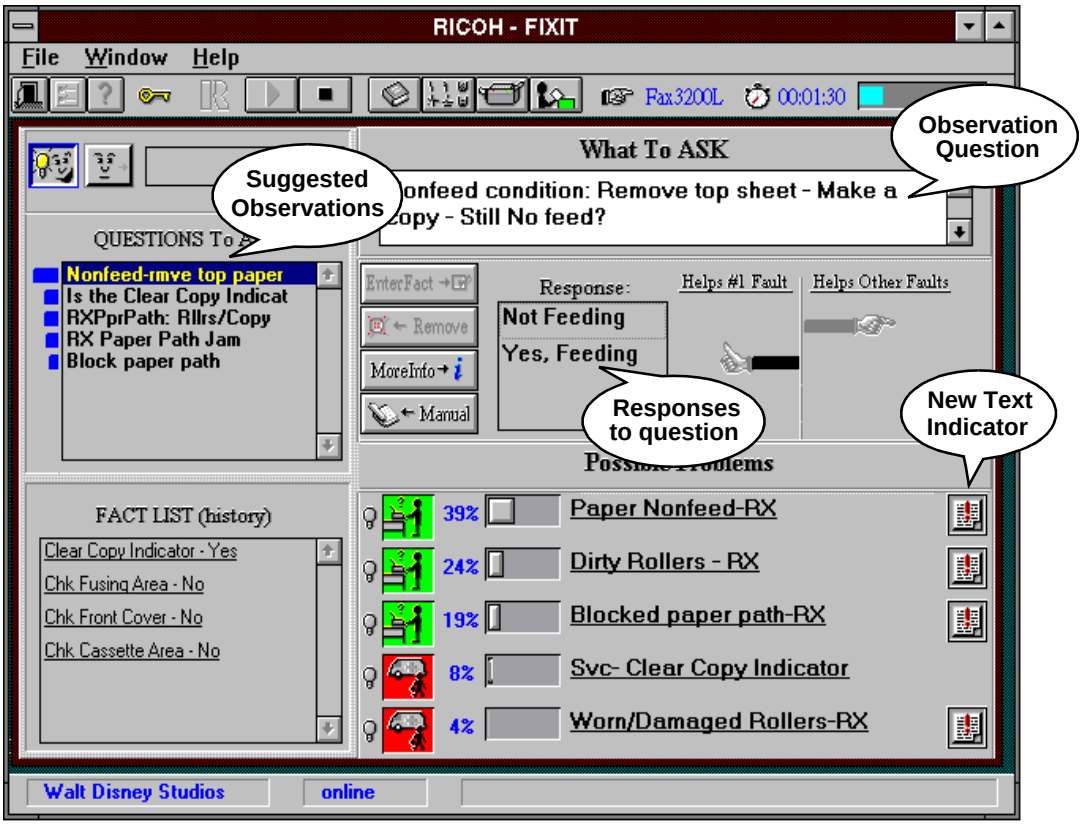

Figure 5a. Agent indicates availability of primary topics

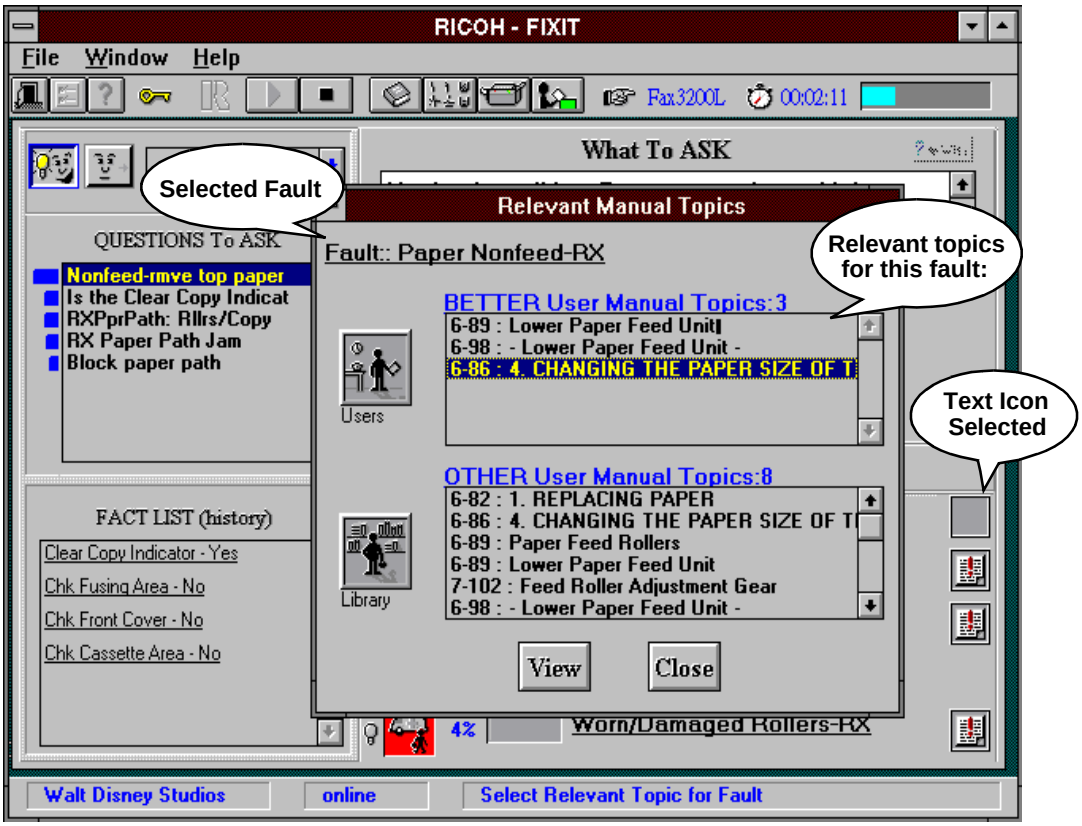

Figure 5b - User selects a primary topic

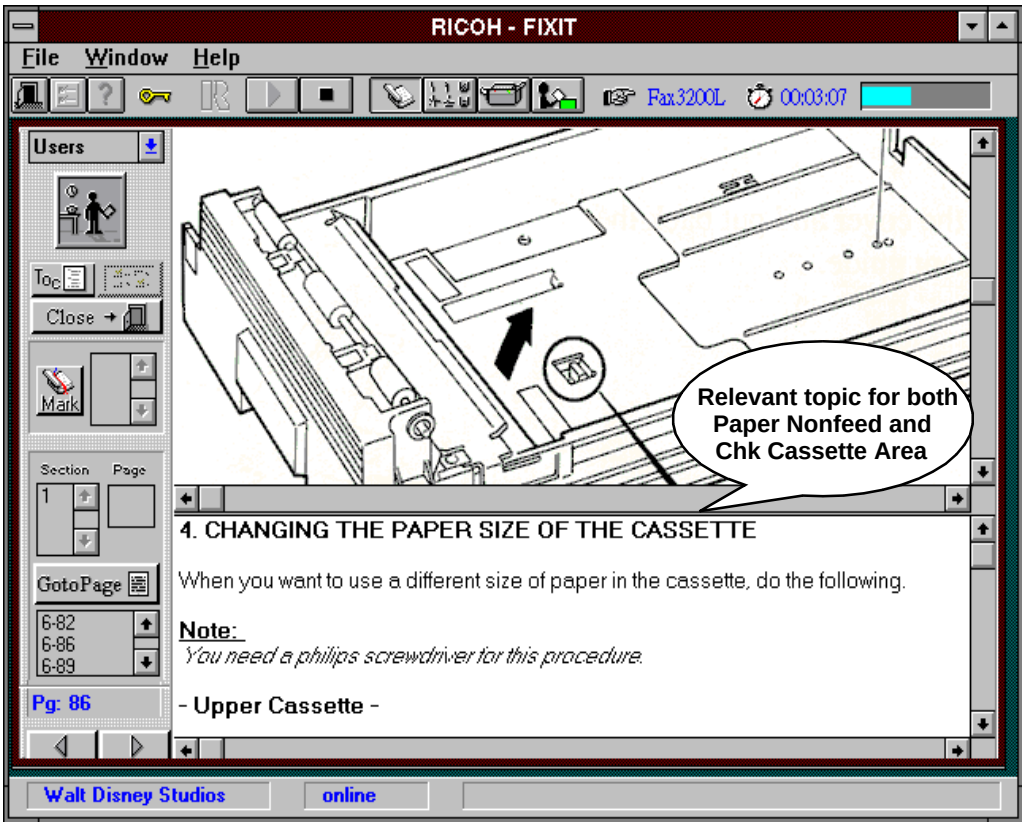

Figure 5c - Documentation browser displaying primary topic

system.

What has been learned from this experience?

Related Work. Decision support systems utilizing multiple technologies can provide fertile ground for exploring new software architectures. As with many developments in this emerging field, the work reported here bears a tantalizing similarity to several other previously-described ideas and yet appears to be different from any of them. For example, work on information filtering might appear to be closely related to ours, but the emphasis there is on a user's "...relatively stable, long-term... goals... that may change slowly over time."[4] This contrasts sharply with the query-free emphasis on contextually-dependent user needs that typically change in the few seconds required to complete a single interaction with the base task. Alternatively, work on plan recognition might in principle support query-free retrieval from external information resources, but the focus there is more typically on generating natural language output from formal internal representations[5].

FIXIT's design philosophy of providing unrequested yet "relevant" information resembles the philosophy underlying the query relaxation approach[6]. There, however, the emphasis is on generalizing a query already posed by a user in order to find "neighboring" information, rather than on avoiding explicit queries entirely. Possibly, the two approaches could be usefully combined.

Related Goals. A common thread running through all these approaches is the goal of providing a user with relevant information beyond that which has been explicitly requested. We believe this to be an important goal because the ever-increasing availability-- and most especially the heterogeneity-- of data, computational and communication resources threatens to overwhelm the abilities of users to access them effectively. This trend has been evident for

many years, but the needs have of course become especially acute with the explosive growth of the World Wide Web. In the context of both the Web as well as in more localized contexts, we should also be thinking about invoking active computation rather than only accessing static databases[7].

Lessons from Fixit. We think the present work provides evidence that the fundamental architecture of FIXIT --namely, a base application which furnishes contextual clues about relevant external information-- is an effective way to provide users with additional useful information that was not explicitly requested. Moreover, the intelligent agent at the core of FIXIT provides the architectural benefit of cleanly separating system components.

At a more detailed level, what can be learned from FIXIT's specific mechanisms? Plainly, the activation predicate and support predicate are tied directly to the belief net formalism. On the other hand, the abstractions underlying these predicates-- focus of attention and support for hypothesis-- have great generality and appeal. The critical assumption underlying their utility is that the user is engaged in a task-oriented dialog whose goals are known or can be determined. While that assumption is not always valid there are of course many application settings in which it is, and in such settings useful if imperfect surrogates for these abstractions may be identified.

Future Research. In the near term, FIXIT might be improved by extending the activation and support predicates, moving beyond the current static thresholds to more refined dynamic ones. More ambitiously, the intelligent agent could be improved to exploit user interactions with the full-text database, rather than with the expert system alone, to assess the current context. Either improvement might lead to greater precision in retrieving database topics.

We think that other exciting opportunities lie in the direction of applying the underlying approach to new application areas. Of these, the Web has been an irresistible magnet for new applications, but at the same time raises difficult interoperability issues at several levels: interoperability across protocol domains[8]; interoperability across pre-existing ontologies[9]; and interoperability by means of cooperative information retrieval agents[10]. In this expansive-- literally global-- context FIXIT's architecture perhaps suggests a solution to one important sub-problem: How can processes for retrieving information from distributed sources be invoked without explicit user request?

While we have developed considerable confidence in FIXIT's architecture, we have no doubt that many challenges lie ahead.

Acknowledgments

We thank our colleagues at Ricoh locations in North America and Japan for their contributions to this work. We also thank the editors and referees of this special issue for suggestions that have improved the presentation of this work.

References

1. D. Heckerman and M.P. Wellman, "Bayesian Networks", *Comm. ACM*, Vol. 38, No. 3, March 1995.

2. J. Graham, "A natural language method of semantic pattern matching for user manual text retrieval", Ricoh California Research Center (CRC) Technical Report CRC-TR-91-16, March 28, 1991.

3. J. Allen, *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, Menlo Park, 1987.

4. N.J. Belkin and W.B. Croft, "Information filtering and information retrieval: two sides of the same coin?", *Comm. ACM*,

Vol. 35, No. 12, 1992.

5. U. Wolz, “Providing opportunistic enrichment in customized on-line assistance”, *Proc. 1993 International Workshop on Intelligent User Interfaces*, Orlando, Florida, January 1993. ACM Press.

6. T. Gaasterland, “Restricting query relaxation through user constraints”, *Int. Conf. on Intelligent and Cooperative Information Systems*, Rotterdam, The Netherlands, May 1993, IEEE Computer Society Press.

7. C. Brown, L. Gasser, D. O’Leary, and A. Sangster, “AI on the WWW: Supply and Demand Agents”, *IEEE Expert/Intelligent Systems & Their Applications*, Vol. 10, No. 4, August 1995.

8. E. Mena, V. Kashyap, A. P. Sheth, A. Illarramendi, “OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies”, *International Conference on Cooperative Information Systems*, Brussels, Belgium, June 1996, IEEE-CS Press, 1996.

9. A. Paepcke, S.B. Cousins, H. Garcia-Molina, S.W.Hassan, S.P. Ketchpel, M. Roscheisen, and T. Winograd, “Using Distributed Objects for Digital Library Interoperability”, *Computer*, Vol. 29, No. 5, May 1996, pp. 61-68.

10. L.F. Bic, M. Fukuda, and M.B. Dillencourt, “Distributed Computing Using Autonomous Objects”, *Computer*, Vol.,No., August 1996, pp. 55-61.