Integrating a Knowledge Base with an open Hypermedia System and its application in an Industrial Environment

Ian Heath[*], Wendy Hall[*], Richard Crowder[#], M.A. Pasha[*] and Paul Soper[*]
[*]Department of Electronics and Computer Science
[#]Department of Electrical Engineering
The University of Southampton
Southampton, UK.
SO17 1BJ
fax: +44 1703 592865
e-mail: ih@ecs.soton.ac.uk

1. Introduction

Logic programming is a powerful framework for supporting reasoning since it is based on predicate logic. It is arguable that the failure of many logic programming (LP) applications to reach industrial strength is due to the difficulties of developing effective interaction with users. Hypermedia systems provide a rich environment within which users can interact with a wide range of document types. Moreover, in recent years the trend is towards implementing *open* hypermedia systems, which allow integration of external applications within a hypermedia framework.

This paper presents the integration of an open hypermedia system, Microcosm, with a knowledge base containing a set of multimedia documents and a rule-base relating to information held in the document set. The architecture of each of the systems is then described, and how they are integrated. Finally, an application of this approach is shown.

2. The Microcosm System

Microcosm[Davis92][Fountain90] is a hypermedia system which allows users to browse and query large multimedia collections. However, Microcosm is fundamentally different to most hypermedia systems because of its ability to integrate information produced using a variety of third-party applications. Microcosm can be seen as an 'umbrella' environment allowing the user to make links from documents in one application to documents in another. Microcosm has been used within the educational field, for large archives, an Urban Information System, for computer aided engineering and for delivering multimedia materials in a range of other applications.

Microcosm consists of a number of autonomous processes which communicate with each other by a message-passing system (Figure 1). No information about links is held in the document data files in the form of mark-up. Instead, all data files remain in the native format of the application that created them. All link information is held in link databases (linkbases), that hold details of the source anchor (if there is one), the destination anchor and any other attributes such as the link description. This model has the advantage that it is possible to have different sets of links for the same data, and also it is possible to make link anchors in documents that are held on read only media such as CD-ROM and videodisc.

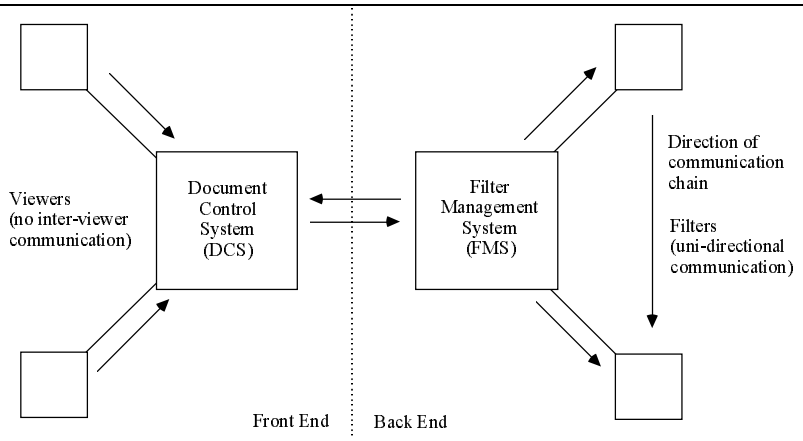

Figure 1: The Microcosm Model

Microcosm allows a number of different actions to be taken on any selected item of interest, so use of the system involves more than simply clicking on buttons to follow links. In Microcosm the user selects the item of interest (e.g. a piece of text) and then chooses an action to take. A button in Microcosm is simply a binding of a specific selection and a particular action.

An important feature of Microcosm is the ability to generalise source anchors. A spectrum of link possibilities exists, from explicitly defined anchors to dynamically generated anchors:

Specific Links. The user is able to follow the link only after selecting the anchor at a specific location in the current document. Specific links may be made into buttons.

Local Links. The user is able to follow the link after selecting the given anchor at any point in the current document.

Generic Links The user is able to follow the link after selecting the given anchor at *any* point in any document. Generic links are of considerable benefit to the author in that a new document may be created and immediately have access to all the generic links that have previously been defined.

Dynamic Links Not all links need to be created by an author. For example, relationships between pieces of information can be computed, based on statistical analysis of the content (e.g. Computed Links), or related information can be retrieved based on attribute coding of documents.

The basic Microcosm processes are viewers and filters.

Viewers Programs which allow the user to view a document in its native format. The task of the viewer is to allow the user to peruse the document, to make selections and to choose actions. Typical actions are *follow link*, *start link* and *end link* (where links may be to processes as well as to documents). The actions themselves are not effected by the viewer. The viewer is responsible for binding the information into a message, which is sent on to the filter chain where it will look for one or more processes that can satisfy this request.

Filters Processes which are responsible for receiving messages, taking any appropriate actions, and then handing the message on to the next filter in the chain. The actions that filters take are such as changing a message, or adding or removing messages.

Because of this flexible and dynamic approach, the Microcosm system is easily extended to provide new functionality. This feature is used exploited to allow the coupling of the system to a knowledge base as described below.

3. Knowledge Based Hypermedia

One way to attack the problem of user interaction within a LP environment is to develop a way of integrating LP as smoothly as possible with other systems written in other languages [Kowalski92]. Preliminary discussion regarding the linking of a LP based system knowledge based system (KBS) with a hypermedia system can be found in[Bench91] and [Soper92].

There are two different approaches to combining a KBS with hypermedia. The first integrates the KBS with the hypermedia system; the second, on the contrary, loosely couples the KBS with the hypermedia system, communication being by message passing. Examples of the first approach are Expertext[Rada89], Locator [Brown91] and PLANTH [Casson92]. Their disadvantage is domain dependence; it is not easy to use a system developed for one domain for another domain without making major changes in the program.

Our KBH framework is an example of the second approach. It has two main advantages: an open architecture and relatively domain independent environment. It aims to improve the performance of both hypermedia systems and KBS's. On the hypermedia side the presence of the KBS can enhance the hypermedia and address some of the fundamental navigational problems. Conversely the presence of the hypermedia can address some of the fundamental problems of user-system interaction associated with current KBS's.

This framework has several advantages:

- The rapidly advancing field of hypermedia management systems can be exploited.
- By using an open architecture approach the strength of the LP system for reasoning can be embedded in the prevailing software environment of hypermedia information retrieval.
- Using hypertext techniques the text can be seen as a high level specification language for some aspect of the LP and KBS. Our answer to the question “how could some experiences of users' requirements be included within a logic programming environment?” is to loosely couple the LP system to the hypermedia system rather than to include these functionality’s in an integrated LP system.

The document base of a hypermedia system and the knowledge base of a KBS are in general domain dependent. The key idea is that the hypermedia manager and the KBS shell can provide a powerful domain independent development environment. We present a general framework for loosely coupling hypermedia and KBS. This framework is implemented for the open hypermedia system Microcosm and a Prolog based KBS shell. Some of the potential of this framework is tested by examples in the industrial domain.

3.1 Interfacing with a Prolog system

The main step in implementing the Knowledge Based Hypermedia (KBH) architecture using Microcosm and Prolog is to provide the means for communication between Microcosm and the Prolog based shell. A new filter KBFILTER has been developed which provides a bi-directional communication link between Microcosm and the Prolog based KBS shell by using the LPA Prolog facility which allows communication with other languages[Steel92]. This filter allows Microcosm to send messages or to receive messages from the KBS shell. Microcosm messages generated for KBFILTER are processed by KBFILTER's message handler. The messages contain necessary contextual information such as the file name from where the message was initiated and the selected text which has a relationship with certain rules in the underlying knowledge base. KBFILTER keeps the file name for further use, translates the selected text to a Prolog acceptable format and passes it on to the KBS shell. The filter uses the Edinburgh syntax of Prolog for communication with KBS shell.

4. An Example: Fault Finding for Pirelli Cables

A cornerstone of a successful industrial maintenance, and engineering policy is to have the ability to integrate many forms of information, from a number of sources, and allow the user to access them with ease. The traditional computer, or paper-based systems used for the presentation and distribution of text and a limited range of graphical information, in a linear or branching fashion are no longer considered sufficient. If the stored information is expanded to include high quality still or animated graphics, and still or moving video, and provided with an access system that permits the user to move easily between these different items of information in a structured fashion, then the management of the engineering data resource for a number of applications can be optimised.

With the increasing complexity of production machinery in the modern automated factory, there is a corresponding increase in the supporting maintenance and production documentation. Industry is therefore looking for advanced informational retrieval systems that can operate both in the office environment, and more importantly on the shop floor. In collaboration with Pirelli Cables at Aberdare, South Wales, a hypermedia system for the provision of process information, maintenance and operator training is being developed, for a sophisticated process machine. The application uses the Microcosm open hypermedia system developed at Southampton. The resource base for this application contains electronic versions of all existing paper documents (i.e. procedures, diagrams and electrical and mechanical drawings), together with additional documents that have been produced specifically for the hypermedia application, these are primarily to allow the user to navigate the resource base.

We want to provide a simple system which allows operators and maintainers to easily and quickly locate information related to a problem with a machine when it fails for some reason. The first approach was to provide a simple question and answer dialogue which allows the KBS (a simple rule-base in this case) to gather information as to the possible cause of a fault. Once the questions have been answered, the operator is presented with a list of possible faults which act as a gateway back into the resource material held within the Microcosm system. To achieve this, we used the LPA Prolog system interfaced to Microcosm via the KBFILTER as described above.

4.1 Rule-Base representation

The rule-base is a simple set of facts, with some rules which provide access to those facts. The only interface between LPA Prolog and the calling application is via ASCII strings. LPA Prolog provides an interface to the Windows environment to allow simple user interaction. However, it was easier to use a separate interface builder (Microsoft's Visual Basic) to build the user interface as a Microcosm viewer. This user interface can communicate with Prolog through the Microcosm system. The Prolog engine can evaluates clauses and return results to the user interface. These results can then be interpreted and displayed in some manner. The Prolog system stores two different types of facts:-

1. Questions which can be used to narrow down the possible faults.

2. Faults, which can occur anywhere in the machine.

Each fault has a set of attributes which define it. These are:-

- A list of tags which describe some unique information that can be used to somehow identify one fault from another. For example, they might describe machine module/physical location (e.g. the list [FF2, Box Folding]).When searching for faults, the engine only looks for those faults which contain the same tags as the ones being searched for. No priority is placed on any of the tags and no ordering is required. It is possible for a fault to be associated with more than one part of the machine. For example, a fault with the tag list [FF2, Box Folding, Flange Inserting], would be found if either [FF2, Box Folding] or [FF2, Flange Inserting] were searched for.

- A textual description of the fault.

- A flag denoting whether the fault is an Operator or Maintainer fault. This flag is used when the fault list is presented so the operator will only see faults that they are able to fix.

- A "probability" or more accurately, "reference count" which provides a simple method for ranking the faults in some sort of order. Each time a fault is identified, this "reference count" is increased. Any list of faults is ordered with the highest "reference count" first. Thus the most common faults will always be the first ones seen.

- A set of strings that can be used as Microcosm generic links to items of related interest. For example, if a fault references certain sensors this section can list the names of these sensors which are then accessible via the user interface component.

- A Prolog clause to be evaluated if this fault is selected. Normally this clause will present the user with a document so that they may identify and correct the problem. However, because this clause is simply a part of the fault, and not evaluated until the fault is chosen, it can perform any task. It might, for example, launch the user into another question and answer session, which might then lead on to another list of faults.

Questions are also represented as facts, although they are far more structured than the faults. A question has a title and a set of possible responses. Each response is described as follows:-

- A list of tags which uniquely identify the question. These are used by the rule-base to locate a particular question and answer set. Unlike faults, these tags do have a relationship between each other. When the rule-base searches for questions, it looks for those questions whose tags exactly match.

- An answer string (e.g. 'The vacuum cups are not working`).

- A Prolog clause that can be evaluated to work out the "reference count" for any faults which can be chosen as a result of selecting this question. This is used to order the possible answers so the most likely one appears first when the list of answers is presented.

- A Prolog clause that can be evaluated to move the user though the fault finding rule-base. At present this term can produce another set of questions (i.e. the designer of the rule base is able to zone in quickly on the possible fault area) or a list of possible faults.

4.2 The User Interface

When the fault finding process is first started, the user interface sends a pre-defined clause which provides the gateway into the rule-base. Currently this returns the question and answers that have an empty tag list associated with them.

This result is then presented to the user as a simple form. The user can select any of the possible answers to the question, at which point the Prolog clause associated with that answer is evaluated. This results in another form being displayed, with another set of questions or possibly a list of possible faults. At any point the user can backtrack through the answers they have given, and choose other answers and explore other possibilities.

The faults act as a gateway into the Microcosm system. When a list of possible faults is displayed, the user is able to follow links from each fault to related information (perhaps giving the location of sensors which need to be tested to determine if that particular fault is occurring). The user can also be directed on to (possibly) other question and answer sessions or other documents in the Microcosm system. This is achieved by pressing the "Tell me more" button. Also, once a fault has been found and corrected, the user presses the "This was the fault" button, which causes the Prolog engine to update the relevant fault's "reference count".

As was mentioned earlier, faults can be flagged as either operator or maintainer faults. If a combination of operator and maintainer faults occur, the maintainer faults are hidden and replaced by a single entry telling the operator to call a maintenance engineer. When such an engineer arrives, they can see all the possible faults by informing the front end that they are a maintainer (by selecting an item on the form).

5. Conclusions

The fault finding system described above is still in its early stages. There is no simple, form based, way of entering new faults or questions. At present, the author has to write Prolog code to create the new facts for the rule-base. However, it should be possible to extend the user interface component so that both faults and questions can be added dynamically. Trials of the system are also needed to test its usability in the factory environment.

Knowledge based hypermedia provides an alternative, and very flexible, way of introducing additional structure into hypermedia. The structure, which may concern a whole range of features from semantic support to text formatting, is modelled in the KB (the contents of the KBS). The relation between the KB and the document base (the contents of the hypermedia) is not part of the system, but it is this informal relation that will underlie the purpose of the particular application being built. The role of the KBS shell is to manage interaction with the KB. This interaction can be initiated when users follow special links, within the hypermedia [Soper92], [Soper94], or by using a specifically tailored front-end as described above. The open architecture means that the KBS component is composable with any suitable present or future hypermedia systems.

6. References

[Bench91] Bench-Capon, T.J.M., Soper, P.J. & Coenen, F. Animation of Advice leaflet Using Hypertext and Knowledge Based system technique. *Proceedings of the Forth International Conference on Law and Computing*, Rotterdam, 1991.

[Steel92] Steel, B. & Westwood, D. LPA 386-PROLOG Programming Guide. *Logic Programming Association Ltd*, 1992.

[Brown91] Brown, P. J. Hypertext: Dreams and Reality. in H. Brown eds. *Hypermedia / Hypertext: An Object-oriented Database*. Chapman and Hall, 1991.

[Casson92] Casson, A. PLANTH: An Expertext Shell for Author and Readers of Technical Documents. *Artificial Intelligence Application Institution*, Technical Data Sheet

[Davis92] Davis, H.C., Hall, W., Heath, I., Hill, G.J. & Wilkins, R.J. Towards an Integrated Environment with Open Hypermedia Systems. *Proceedings of the ACM Conference on Hypertext*, ECHT'92, Milan, Italy, December 1992, pp 181-190. ACM Press 1992.

[Fountain90] Fountain, A.M., Hall, W., Heath, I. & Davis, H.C. MICROCOSM: An Open Model for Hypermedia with Dynamic Linking, in A. Rizk, N. Streitz & J. Andre *eds. Hypertext: Concepts, Systems and Applications. The Proceedings of the European Conference on Hypertext,* INRIA, France, pp 298-311 Cambridge University Press. 1990

[Kowalski92] Kowalski, R. *FGCS-A Person Perspective*, Department of Computer Science, Imperial College, 1992.

[Rada89] Rada, R. & Barlow, J. Expertext: Expert System and Hypertext. *Proceedings of the EXSYS*, IITT International, 1989.

[Soper92] Soper, P.J & Bench-Capon, T.J.M. Using Hypertext to Interface to Legal Knowledge Based Systems in *Proceedings of the DEXA 92* Valencia, Springer-Verlag, 1992.

[Soper94] Soper, P. & Pasha, M.A. KBH: A Framework For Legal Information Management. in *Conference proceeding of ITA'94*, 1994.