The Design and Implementation of an Open Hypermedia System

Wendy Hall
Ian Heath
Gary Hill
Hugh Davis
Rob Wilkins

CSTR 92-19

University of Southampton

Department of Electronics and Computer Science
University of Southampton
Southampton SO9 5NH.

Abstract

This paper describes the design and implementation of Microcosm, an open hypermedia system. It examines the limitations of the initial prototype and describes the redesign that resulted in Microcosm version 2.0,.as well as details of its implementation. The new model has a greatly enhanced filter mechanism for handling the back-end processes of the system, such as link database access. The filter system can be easily extended and used to implement new functions within Microcosm. The paper describes how this capability has been utilised to provide a simple set of navigation tools. Finally the paper explores the implications of the model and potential future enhancements.

Contents

1. Introduction....2
2. Microcosm - An Open Model for Hypermedia....2
3. The Design of Microcosm....3
   3.1. Microcosm version 1.0....4
   3.2. Microcosm version 2.0....5
      3.2.1. A New Message System....5
      3.2.2. The Notion of Typed Documents....5
      3.2.3. Action Configurability....5
      3.2.4. Expansion of the Linkbase System....6
      3.2.5. A Multi-User Hypertext System....6
      3.2.6. An Overview of the New Design....7
4. Implementation Details....7
   4.1. Microcosm Messages....7
   4.2. The Communications System....8
   4.3. The Document Control System....8
   4.4. The Filter Management System....8
   4.5. Microcosm Filters....9
5. Implications of the Model....10
6. Navigation tools - extending the filter chain.....10
7. The Future....12
   7.1. New Filter Architectures.....12
   7.2. Networking....13
   7.3. Other implementations.....13
   7.4. Navigation Device integration.....13
8. Conclusions....14
Bibliography....14

Introduction

This paper describes the current state of development of the open hypermedia system, Microcosm. This includes a description of the filter management system which was added to the model after the initial prototyping, implementation details, and a discussion of current research work being undertaken using the Microcosm model.

Microcosm - An Open Model for Hypermedia

The key motivation behind the Microcosm project is to produce a fully open and user-configurable workbench for research into hypermedia. Research into the desirable features of such a system identified the following problems with the majority of existing hypermedia systems.

Authoring effort required. Generally links need to be individually created and all documents must be linked into a hypermedia context before they can be useful. The large effort required to insert links into documents is seen as one of the main limitations of hypertext systems.

Closed systems. Current systems generally run as a stand-alone program with only unidirectional communication capability with other packages. Extensions to systems to allow video and graphics capabilities tend also to be unidirectional and so extended media types cannot function fully as a part of the system.

Proprietary document formats. Documents produced for one system are not easily used in others.

Linking from read-only media. Generally links are embedded in documents. This causes problems when linking from and to read-only files (for example on a CD-ROM) without making copies of files which may be very large.

Although not all systems suffer from all of these problems, the aim with Microcosm was to design a system that addressed them all. To guide the system design, the following principles were used to evaluate alternative solutions.

Separation of links from data. The relationships between documents are separated from the actual documents, allowing them to be analysed, processed and manipulated in their own right.

Loosely coupled system with minimal interdependencies. The system is not just one program, but a set of communicating tasks. Additional tasks can therefore be added to the system or existing tasks replaced.

No ultimate distinction between author and user. All users of the system are in a position to add links.

A full explanation of the system objectives and the solutions to the problems raised by the principles described above was first presented in (Fountain et al 1990).

The Design of Microcosm

Research into the desirable features of an open hypermedia system based on the principles outlined above involved a survey and analysis of existing hypertext/hypermedia models such as Guide (Brown 1987), Notecards (Halasz 1988), Intermedia (Yankelovich et al 1988), KMS (Akscyn et al 1988), and Sun's Link Service (Pearl 1989). The keys to the design of Microcosm were the separation of links from data objects into a link database, or linkbase, and the identification of the selection/action model as the primary means of user interaction with the system. The user selects information (for example a text string) and chooses some action to be performed upon it. The actions provided by the system need not be limited to those which are traditionally associated with hypertext systems: they can be diverse as 'look up in glossary' or 'perform some image processing operation'.

Link following becomes just a single action within the system. It is achieved in exactly the same way as any other action, namely by selecting the start anchor of the link and choosing a particular action such as follow link. Creation of links can be implemented in the same way. Buttons can be modelled easily within this environment: a button is simply the binding together of a particular selection and the action 'follow link'. The selection/action model provides a very open framework within which a hypertext system can be built. There is no upper bound on the functionality that can be supplied, as new features can be added by means of the new actions that the system supports.

In Microcosm, all link following is resolved by a simple database lookup given the item selected and its position in the document. The function of a button in the Microcosm model is therefore to launch a pre-defined linkbase query. The author of the button will usually ensure that this query will find a particular link (or set of links). As far as the user is concerned, the button (which is highlighted when the document is displayed) will function in the same way as a conventional button in a hypertext system.

However, in Microcosm, the user may select any word or phrase (whether indicated to be a button or otherwise) and invoke the follow link command Microcosm will then perform a linkbase query on the selection and inform the user if any links exist for the given selection. Links can be defined as only applying to a particular location in a particular document (in which case the offset within the file is stored in the linkbase and the link selection can be made into a button), or applying to any occurrence of the selection in a particular document, or applying to any occurrence of the selection in any document in the system. Since the latter case represents the most general form of link following, we call such links generic links. The implications of generic links are discussed later in this paper.

This model can easily be extended to incorporate link following and link creation from media types other than text, although the concept of a generic link can only be realistically applied to text documents within the bounds of current technology.

Microcosm version 1.0

Microcosm version 1.0 (Fountain et al 1990) was prototyped in Microsoft Windows version 2.11, and fully implemented in Windows 3.0 after the latter's release. It consisted of a document control system (DCS) that managed the document viewers, routed messsages and supported the linking mechanism. In keeping with the modular design, different viewers were used for different media types, so that each viewer is a separate module within the system providing the interactive component required for a partiuclar media type (text,

graphics, audio, video etc.). This gave the necessary flexibility for an open system along with the 'selection action' model.

Document viewers communicate by means of messages, and all message are routed through the DCS. In Microcosm 1.0 messages consisted of three parts: a selection, an action and context information (such as the name of the document containing the selection). Messages could be created in two ways: either by the user selecting an item on the screen and choosing an action from the pull-down menu bar, or by a button being selected. Context information was provided by the document viewer. Communication of messages between the viewers and the DCS was handled using the Dynamic Data Exchange (DDE) protocols provided by Microsoft Windows with an additional library of channel functions to make the use of the DDE as transparent as possible.

In Microcosm 1.0, links were stored in a simple flat-file database with one record per link. Each record associates a link message with a dispatch message. When the database receives a follow link action, it attempts to match the rest of the message with a link record in the database. If a match is found, the dispatch message is sent back to the DCS, and this information is used to startup the document viewer of the appropriate type. The user is unaware of the operation of the DCS. From their point of view, they have asked to follow a link and the new document has appeared in another window. If more than one link is defined on the user's selection, then a dialogue box presents the user with a choice of documents to view.

Microcosm 1.0 demonstrated the feasibility of the principle objectives of the Microcosm model. However, this first implementation of the model showed up many flaws in the original design and many ways in which it could be improved. The linkbase and message passing functionality was fixed and fairly limited, and whilst operating on the principle of parallel communicating processes, the separate processes were bound rather tightly together and the system modularity was rather limited. The redesign resulted in Microcosm version 2.0, which is described below.

Microcosm version 2.0

A New Message System

The initial design of the message structure for Microcosm 1.0 was fixed. Each message contained fields for all the information required to describe the source point of a link, the selection, and an action. However, the fixed nature of the message structure would require modification to the system each and every time a new field was added to a message. This was undesirable and so a more flexible message format for Microcosm was designed.

As the number and type of fields that might be required in the future could not be predicted, it was necessary to design and implement a totally flexible message format, capable of storing any set of fields required and being of any size. These requirements meant there would have to be a set of functions available to manage these new messages as they have moved from being static to dynamic objects. However, the same approach could be applied to this area as was applied to the channel communications system implemented for the original prototype. A set of functions could be designed and then implemented as a DLL, providing a consistent interface to the message regardless of the underlying storage strategy employed.

The Notion of Typed Documents

The original system used the extension of a file to determine the type of the document and which viewer to use to display it. This approach was very limiting. It is generally the case that users group files by using extensions on IBM-PC machines, as the three letter suffixes are useful for this. On other systems where there are larger limits to the size of a filename and no inherent format, this is not so prevalent. It would be better if the system had an understanding of a document type separate from the document file name, and used this type to decide how a document is to be treated. Thus the types of the document containing the source and destination anchors would be stored as part of the link information, and passed back into the system when a link is followed and used by the system to determine the correct viewer to use.

Action Configurability

The original design of Microcosm had any and all action processing carried out in the DCS. However, this design was not configurable from the user's point of view. The solution came from looking at the design of the document viewer part of the system. Here, we have a set of processes, each one responsible for a specific task within Microcosm, namely the display and interaction of a particular type of document. Managing these views is a controller, which makes decisions about what viewers are to be used and maintaining communication between them. This design could easily be adapted to provide a simple model for action processing. Each action could be handled by a separate process and some action management system could look after these components.

The behaviour of the components can be thought of like filters, and this was the name given to them. In this way Microcosm was split into two co-operating processes. Firstly, the Document Control System (DCS) which is responsible for managing the documents which make up the front-end as before. Secondly, a back-end system responsible for managing all the action processes, the Filter Management System (FMS).

Expansion of the Linkbase System

Microcosm 1.0 employed a single link database, or linkbase that was present for each and every invocation of the system. An obvious improvement was to allow the user to specify a database to be used when the system is started. This would allow the user to maintain different linkbases containing links which it was felt should be kept together. However, this approach still has failings. If there were several of these linkbases which the user wished to use in a particular session, the user would have to incorporate all of the separate linkbases into one meta-linkbase again using some tool to achieve this aim.

The solution came from two areas, namely the concept that link following/creation is a single action of the system, and the design work already carried out to make the action processing aspect of Microcosm more configurable. Instead of having just one linkbase in the system, we could have as many as was deemed necessary. By limiting the link creation to just working on one of the linkbases, perhaps the first one encountered by the system, the area of effect of link making could be reduced to a single linkbase under the user's control. Also, many link-sets could be overlaid on to a single set of documents at the same time, perhaps providing links which the user would not have had access to otherwise.

With there being no limit except those imposed by the implementation, it could be that the linkbases would provide thousands of links for one particular follow link query. This would result in information overload. However, if contextual information is stored along with the link it may be possible to limit the amount of links the user has to deal with. The aim was to provide filters which would look at all of the links generated by the linkbases and selectively filter out those links deemed unsuitable or unnecessary for the user perhaps by some user-selectable criteria. The open model that Microcosm provides is eminently suited to this task, as action processes may be added to the system.

A Multi-User Hypertext System

A consideration of hypertext systems currently is how to support the notion of users in the system, and perhaps how to allow users to work together in some collaborative manner. There were two areas to consider. Firstly, there was the scenario where more than one user could use a particular installation of Microcosm, although not in a multi-user mode of operation. It was decided to expand the linkbase notion slightly and give each user their own private linkbase. It would be here that any links made would be stored by default. Other linkbases could be used at the same time to provide other sets of links, and tools would allow links to be moved between linkbases. Microcosm would thus have to understand about different users. A logon sequence would be added to the start-up sequence for the application.

The second area to consider was a networked version of the hypertext system. The underlying model makes no assumptions as to where data and applications are located and so should not preclude a network topology for the system, at least allowing remote access to data/applications. However, further work will have to be undertaken to examine what extensions to the model are required to provide a true multi-user system.

An Overview of the New Design

The new design for Microcosm outlined above is summarised in the new conceptual model shown in figure 1. There is now a symmetry to the design, with the action management system mirroring the document management system. The details of the implementation that follow are taken from (Heath 1992). A detailed discussion of the design and implementation of the Microcosm linkbase is given in (Heath, 1992).

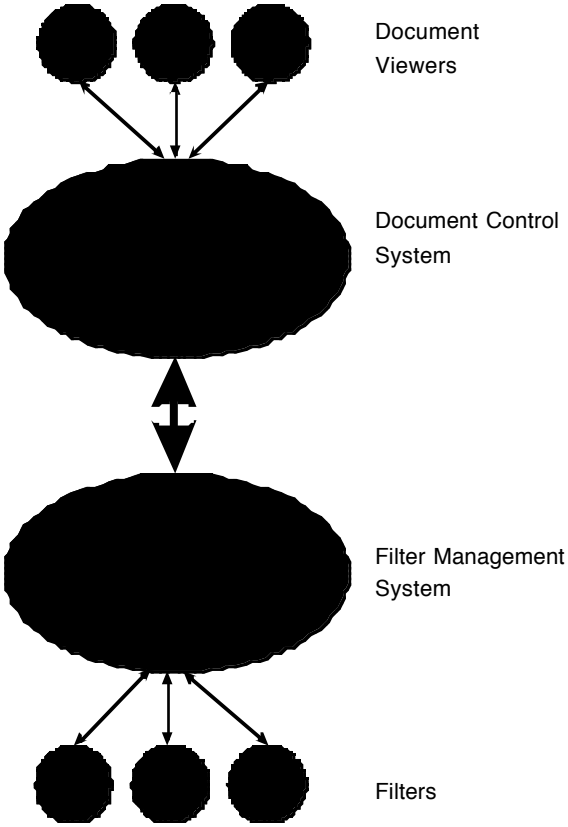

Figure 1: Microcosm Process Model

Implementation Details

Microcosm Messages

Microcosm 2.0 treats the message as an object. The various parts of the system receive these objects, and have access to a set of routines that can manipulate them. Like the channel system, the underlying method by which an object is physically implemented is hidden away from the applications, all access is performed using a message library. Terminology from markup languages was used as this would closely mirror the initial implementation and also was a good way of visualising these message objects. A message contains a set of fields called tags. Each tag is comprised of two parts, a tag name and a tag body. A tag name is a simple string uniquely identifying the tag, and the tag body contains whatever information is required. A set of functions were defined that allow full access to the message object.

A Microcosm message object is simply a string containing a set of tags, each one representing a single field in the message. The tag-body may contain any data, and is specially encoded so as to remove any ambiguity that may occur should the tag 'escape' character, currently a backslash, appear in the body of the message.

The Communications System

The communications system and its channel abstraction was retained from Microcosm 1.0. However due to the increased modularity of the new model, much more message traffic was expected requiring attention to the handling

of intensive message passing without losing messages. The underlying message passing was also rewritten to use the underlying Windows message system rather than DDE, for speed reasons.

Where communication with existing Windows applications is required using a DDE connection, a small bridging application is used to accept messages from the Microcosm message system, translate the Microcosm message into the form required by the particular application, and send it to the application using DDE.

The Document Control System

The implementation of the DCS was similar to the original, with improvements made to handle the new messages and the action processing component removed. Starting up the DCS causes the rest of Microcosm to load. In fact, the intialisation task is shared between the DCS and the Filter Manager. The DCS knows which application to start up, and the Filter Manager takes care of starting up the back-end components. The DCS, as before, manages all communications between the viewers and Microcosm 2.0. It now, however, passes all action commands to the Filter Manager. Any messages which are returned to the DCS are interpreted and may cause new viewers to start up. In this way the action processing is performed in a totally separate manner to the document management.

The Filter Management System

It is through the Filter Management System (FMS) that the functionality of Microcosm is mainly achieved. This application is responsible for providing the functionality of the system by communicating with the individual filters.

The design chosen was that of a chain of filters. Messages originate from the DCS, are passed into the first filter on the chain, then to the second and so on. Once the message has passed through the last filter, it is returned to the DCS. The initial implementation split this chain into two sections, a 'linker' chain that would find and create links and a 'service' chain that would control and filter the links generated by the linker chain. Action messages such as 'Follow Link' were synchronised by the use of control messages that filters sent to indicate that they had finished processing the current action message. This enabled the FMS to detect when an action message had been processed by all filters in the linker chain. When the linker chain filters had finished, a control message was sent to the 'service' chain to inform the other filters that the action message had been fully processed. Synchronisation is required by some filters, for example the Available Links dialogue, in order to detect when all processing has finished.

This model worked, but with use, it became apparent that it was overcomplicated both for message passing and from a user viewpoint. The model was therefore simplified, with a single chain for all types of filter. Filter synchronisation was initially carried out in a similar way, but the 'filter completed' message became a form of token that moved along the filter chain as the filters completed processing. The drawback of this was that filters always had to keep track of whether they held the token and send whether they were processing the action message or not. To reduce this complexity in the filters, the synchronisation method was changed slightly so that only filters that wanted to process the current action message would produce control messages. On receipt of an action message that it wishes to process, the filter sends out a 'processing' message. Then when it finishes processing, a symmetrical 'completed' message is generated. By analysing the flow of these

control message pairs, a filter that requires synchronisation can detect when the filters that precede it have finished processing the current action.

The FMS has three main tasks to perform in maintaining the model described above. Firstly, to start up all of the user's filters as described in the user's configuration file. Secondly, to manage message routing through the filter chain while Microcosm is running. Finally, the FMS allows the filter configuration to change dynamically. A dialog box presents the user with a list of the currently active filters. By selecting a particular filter, the user is able to relocate its position in the chain, perhaps reducing the effect of the filter, or even closing it down altogether. The user can also add new filters into the system to provide added functionality. To do this, the user selects the filter from the list provided and positions it in the chain. The list of available filters is again kept in the user's configuration file.

Microcosm Filters

The way in which filters work in Microcosm 2.0 closely mirrors the way document viewers work. A filter, when invoked, opens a communications channel with Microcosm, and through it sends and receives messages. In order for it to work properly in the chain topology, a filter has to send back any messages it does not wish to process, so that the next filter in the chain can process it. If a filter wishes to block a message it can simply do this by not returning it. To process a message the filter simply performs the relevant task, and then sends any messages generated from this back to the FMS and thus into Microcosm. In addition the filter may generate synchronisation messages as described above, in order to allow other filters to keep track of message processing activity.

Implications of the Model

The separation of links into a separate knowledge base, and the resulting ability to define generic links, greatly reduce the authoring effort in creating hypertext systems. For example, suppose an author wishes to make some link on the phrase tropical rainforest. It may be that a simple definition is required and the author wishes the user to be able to access it from anywhere that the phrase appears. Defining this link, once, as a generic link makes this possible. In other systems, the author would have to create some sort of link from every occurrence of the word, or set up an automatic link creation program to perform the same function. This would have to be up-dated everytime a new document was imported into the system. In Microcosm, generic links apply automatically to any new documents. The separation of links also addresses the problem of read-only media. Generic links can apply to any files from a CD-ROM in the same way as to any other files and other links can also be made from the CD-ROM file, and details stored in the separate linkbase.

Using the Dynamic Data Exchange (DDE) capabilities of Windows 3, we are able to extend the generic link facility to any application running in that same environment. Applications that we have integrated into Microcosm in this way include Guide, Toolbook, Superbase, Word and the geographical information systems SPANS. In theory, there is no limit to the number or type of application that can be integrated in this way.

Navigation tools - extending the filter chain.

The back end of the Microcosm model (i.e. the filter chain) can be easily extended as noted above. Any such extension will allow the user to add or include navigation tools in their customised system. The filter chain is the perfect placing for any such navigation device as any information it may need, for example which documents are being viewed, will pass along it during the users interaction with the system. If the navigation device wished to mimic the user in any way, for example if it were to take the user through a guided tour or backtrack to a previously viewed document, then the system would not need to be able to differentiate between a device or user initiated action. Due to the equally flexible messaging system implemented within the Microcosm model any information that a filter may need that is not already available can simply be generated by, either creating a completely new message or adding extra tags to an appropriate message that already exists.

The type of navigation devices that it was chosen to implement were largely dictated by the Microcosm model. It made the implementation of any type of dynamic map impractical for the following reasons :-

1. The ability to view any document via any Windows program meant that the domain of the map was never complete.

2. The ability to define generic links, whose starting points are effectively unconstrained until selected by the user, means that the links on the map cannot easily be defined without a considerable amount of precomputation.

3. The separation of the link layer from the document layer means that the map devices would have to carry out a large amount of linkbase analysis in order to build up any kind of map.

If a map device was to be constructed it would have to be able to compute the map off-line; in most cases this would be unsatisfactory as the map would be out of date and incomplete.

It was for these reasons that the first navigation device to be written was a history type device. This records the documents that the user has viewed in chronological order. The user can, at any time pull up a history list of their current session. Using this they can re-open any previously viewed document. They are also provided with a short cut to retrace their steps from document to document back through their history. Later extensions to the message system allow the history device to show whether the documents in the history list are currently open on the desktop. This could save time if the user wished to see a document, such as a video sequence, which takes time for the system to access and prepare.

The second navigation device to be implemented was a device to allow the user to construct and view guided tours or mimics as they are called within the Microcosm system. This task was split into three separate devices :-

1. Mimic viewer, provides the front end interface for the user to the mimic system.

2. Mimic engine, provides the back end 'work-horse' for the mimic system which sits in the filter chain.

3. Mimic generator, provides the user with a tool to create and convert between different forms of mimic.

The user is given three main methods for generating mimics, a text based language, a form based creation tool and the history files saved from the history device. The mimic generator allows for the easy interchange between any of these three methods. The mimic has the same level of control over the document side of Microcosm as the user. The mimic can contain branching, sequential or parallel portions, the documents can be launched by the system, clock or user and mimics can be created that are completely void of any user interaction, effectively a presentation.

The third navigation device is the short term memory device, this is a plug-in replacement for the available links device. If the user chooses a link that has multiple end points the list of end points is shown to the user in the short term memory device or the available links box. The short term memory shows the end points of selected links and uses colour to give the user some idea of how long ago they had viewed the resulting documents, if indeed they had viewed them. It also removes any duplicate entries in the list presented to the user. These two enhancements will reduce the cognitive load associated with presenting the user with multiple end points from one link etc. The user has full control of the memory length and other such useful settings.

The fourth navigation device is the showlinks device, this provides a working solution to the problems involved with the user being able to locate generic links within documents. It will analyse the current linkbases and show the user any generic links that occur within the currently selected section of the document. This device has been optimised to be as quick as possible, as a result of this optimisation it will only look for generic links that are one or two words in length. This will typically provide the user with 80% or more of the generic links that would be found should the showlinks device carry out a full linkbase analysis.

The Future

New Filter Architectures.

Whilst the current filter architecture of a simple chain works well, we have often encountered situations where its simplicity is restrictive. In particular, its linear nature makes the ordering of filters very important and can preclude some filters from working together due to their placement requirements. Also from a user viewpoint, it is necessary to have some understanding of the interaction between various available filters before reordering or adding tasks. Without this understanding it is possible to cause the system to cease working as expected. This is satisfactory for experienced users, but a naive user could soon create problems for themselves.

Similarly, from an implementation viewpoint, the simple chain model can restrict the capabilities that can be incorporated into filters and is also inefficient in that all messages are routed through all active filters. A typical example would be a viewer that wishes to send a message to a particular filter, this message would be sent through to the FMS which would simply send it on to the start of the chain and help it to trickle through all the filters, whereas a more efficient solution would be for the FMS to establish where the message needs to go and send it directly to the correct filter or set of filters. Alternatively, consider a mapping filter that attempts to build a local map of links around a specified document. This would need to send queries to the active linkbases and have the results sent back to it, which is not possible with the current model.

A more friendly system would allow the user to simply specify a new filter to run, and would take care of its placement in the set of active filters (not necessarily a chain) automatically. Possible architectures might allow dynamic routing of messages through the filters according to the action required, grouping of filters into classes according to their functionality, and communication between filters.

Networking.

Networking is a key aspect of future development that will make Microcosm a much more versatile environment, where authors on different machines can cooperate in authoring specific information sets and users can access links and documents stored on remote machines transparently. This will help change the emphasis of the system from presentation and delivery of specific information sets towards sharing of information and cooperation between users of the system.

As described earlier, effective development of these features is tied into development of a more versatile filter architecture as described in the previous section. This is because the chain model is biased towards the current user's requirements, although it consists of separate filter processes, it is difficult to address particular filters, or to make only certain parts of the system available to the network. This means the model is unlikely to translate well to requests made by remote users.

Another important aspect is the development of a document database or 'docuverse' that will aid in the location and retrieval of documents in a distributed environment where files may move or become unavailable due to network failures, as well as finding relevant documents using a more traditional keyword system.

Other implementations.

The Microcosm model is also being implemented on the Apple Macintosh and UNIX platforms. The separation of the linking information from the resource base should enable us to easily share information between platforms. With an extended model incorporating distributed links and documents, it will be possible to maintain a heterogeneous system comprising a variety of different architectures all interacting transparently.

Navigation Device integration.

The devices already implemented are, to a certain extent, integrated with each other. For example the user is able to use the history device to create mimics via the mimic generator. Future integration would allow other navigation devices to request information about a document from the history and memory devices. A map device could use information from the memory device, should it be active, to show the user whether a document on the map has been seen recently. The history device could be asked whether the document in question is currently open on the desktop, this too could be shown on the map.

The integration of the navigation devices should make them more powerful and more useful to the user. Again because of the flexibility of the Microcosm message and filter chain system the above inter-device interaction should present no real problems. It will be a simple case of adding new messages to the already existing message set and making the relevant parts of the filter chain aware of them. New types of navigation tools need to be developed to cope with the added complexities of open systems. The design of Microcosm is ideal for this type of experimentation.

Conclusions

The need for open hypermedia systems has been well-argued by various authors (Fountain, 1990, Halasz, 1991, Malcolm et al, 1991). This paper has described the design and implementation of one possible model. The keys to this model are the separation of linking information from source material, and the object-oriented approach to its implementation as a set of parallel communicating processes. The flexibility thus derived allows new functionality to be added to the system with relative ease. This creates a workbench for hypermedia research as well as enabling users to easily customise the system to their own needs.

As international standards emerge for hypermedia and multimedia systems, they can be incorporated into the model as new modules without affecting the underlying functionality of the system. However, most importantly, Microcosm provides a model of open hypermedia that allows other applications, such as databases, spreadsheets and word-processors, to have a hypermedia functionality, so that data within them can be linked through appropriate linkbases to common sets of multimedia information and documents. This enables users to operate in an integrated information environment, using standard processing tools with a hypermedia capability. Microcosm is therefore not just another hypermedia authoring or presentation system. It is more an extension to the operating system, providing a link service to other applications within that same environment, and potentially in a distributed, multi-platform environment as well. This is where the future for hypermedia lies, and where future research in to the Microcosm model will be directed.

Bibliography

Akscyn, R.M., McCracken, D.L. & Yoder, E.A. "KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations". Proceedings of the ACM, Vol 13, pp820-835, July 1988

Brown, P. "Turning Ideas into Products: The Guide System". Hypertext '87 Papers, pp33-40, Nov 1987.

Fountain, A, Hall, W., Heath, I & Davis, H. "Microcosm: An Open Model for Hypermedia with Dynamic Linking". Proceedings of ECHT'90, pp 298-311, Nov 1990.

Halasz F.G. "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems". Communications of the ACM, 31(7), July 1988, pp 836-852.

Halasz F. G., (1991) Seven issues: Revisited. Hypertext '91, Closing keynote address.

Heath I., (1992), An Open Model for Hypermedia: Abstracting Links from Documents, Ph.D. Thesis, University of Southampton, UK, 1992. (In Preparation).

Malcolm K. C., Poltrock S. E. & Schuler D., (1991) Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise. In Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX Dec. 15-18) 13 - 24.

Nielsen J. "The Art of Navigating Through Hypertext". Communications of the ACM, 33(3), March 1990, pp 297-310.

Conklin J. "Hypertext: An Introduction and Survey". IEEE Computer, September 1987, pp 17-41.

Pearl, A. "SUN's LInk Service: A Protocol for Open Linking". Hypertext'89 Proceedings, pp137-146, Nov 1989.

Yankelovich N., Haan B., Meyrowitz N. & Drucker S. "Intermedia: The Concept and Construction of a Seamless Information Environment". IEEE Computer, January 1988, pp 81-96.