Towards An Integrated Information Environment With Open Hypermedia Systems.

Hugh Davis Wendy Hall Ian Heath Gary Hill
Rob Wilkins.
The Department of Electronics and Computer Science,
The University of Southampton, Southampton, SO9 5NH, U.K.

October 11, 1993

Abstract

This paper examines open hypermedia systems, and presents the case that such systems provide a step forward for dealing with large, dynamic data sets in distributed, heterogeneous environments by allowing users to access and integrate information and processes in richer and more diverse ways than has previously been possible. In particular, the enhanced Microcosm model for open hypermedia is examined, and the ways in which it provides such an environment are discussed. The paper continues by investigating the advantages and the shortcomings of this model and identifies the areas in which further work must be completed before such systems can become widely adopted, such as the granularity of source and destination anchors, editing and version control. Some attempts to provide solutions to these problems are presented and discussed.

Keywords

open, hypermedia, integration, Microcosm

1 Introduction

In their seminal paper at Hypertext '91 Malcolm et al.[9] outlined the reasons why hypermedia is not yet satisfactorily usable for a large engineering enterprise and at the same conference Frank Halasz gave the keynote talk [5] in which he revisited his "seven issues" [3]. These papers point the way towards third generation hypermedia and Halasz[5] charts the progress that has been made so far in this direction.

At the University of Southampton we have been working since 1989 on a model for open hypermedia, and have produced a system called Microcosm [2]. This system was initially designed with the intention of providing a testbed on which the research team would be able to experiment with various ideas in the field of multimedia, but is now being used at a number of sites for integrating multimedia applications and delivery of research and teaching materials. The open perspective of the system is the main attraction of the product. However, the authors believe that using a hypermedia system as a presentation platform for multimedia information is simply a partial solution to a larger problem, which is the provision of a complete hypermedia environment at operating system level. In this paper we examine the progress of open hypermedia systems towards providing "industrial strength hypermedia".

Many people use PC's and workstations as regular tools in their working day. What do they do with these machines? On the whole they use a relatively small number of packages to process data in some way that is pertinent to their job, and the remainder of the time they use the machine as a kind of electronic log book. They prepare documents, they send and receive electronic mail, they save documents they have received, they keep calendars and "to-do" lists, and they keep notes about tasks in which they are involved. All these activities produce a large amount of data which is kept on the machine. How do they navigate through this data? Most users survive by using the file directory structure as the primary method of locating data, and perhaps some utility such as *grep* to help locate relevant files.

If hypermedia techniques for linking and navigating information are so effective, why are users not using these systems for integrating their data? We suggest, as did Malcolm et al , that the answer to this question is that current systems do not have the properties to enable this facility: the cost of attempting to use current hypermedia systems in such a dynamic way is higher than the benefit gained by the added value. In order to make use of a document in a hypermedia system, it must generally first be *imported* to the system. As soon as links are added to the system the data to represent the link will be stored in the file as some form of mark-up. The data is now in a closed system; it is no longer possible to process the data with the package that created it, as it is stored in a format private to the hypermedia system. If the system allows the user to export the data again, it will certainly lose the link mark-up information. In any case where data is dynamic in nature it seems unlikely that users will be prepared to risk this process of "handing over" their data, and furthermore it seems that the manual effort involved is likely to deter such behaviour unless the user is fairly certain of the need for the extra facilities provided by the hypermedia system.

While hypermedia systems remain closed they are likely to continue to be confined to applications where the data remains relatively static. The next generation of hypermedia must appear to the user as a facility of the operating system that is permanently available to add information linking and navigation facilities with the minimum amount of user intervention and without subtracting any of the functionality that was previously available. They must be able to make links to and from data produced by other information processing tools, and must have tools for automatic link generation and maintenance. Sun's Link Service [12] and Microcosm [2] are examples of hypermedia systems that have attempted to address these problems.

2 What Users Want from a System

The paper "Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise" [9] defines user requirements for hypermedia that extend well beyond those of an engineer. In this section we summarise these general requirements.

- An adaptable environment for the integration of data, tools and services.

  Closed systems that keep data in a private format must also provide tools to access and process that data. e.g. Intermedia [14] provides applications such as InterText and InterDraw. However, this approach is a closed solution. Arguments amongst users about choice of text editors and drawing packages become almost religious in intensity. Users do not want to be confined to a particular package, and in any case it is not possible to predict the facilities that all users will require, or sensible to attempt to compete with the major software houses in replacing the functionality that they provide in their packages. Hypermedia systems must allow users to create data in whatever package they choose, and then to make links from data to data, or to make active anchors [11] in these applications.

- A System which is platform independent and distributed across platforms.

  Users will wish to link from data and applications on one machine to data and applications on other machines. They will want these operations to be transparent. In most respects this is an issue in the domain of distributed operating systems. However, the hypermedia functionality must be distributed across the operating system, and link information must be portable across hardware platforms.

- A system which enables users to easily find, update, annotate and exchange information.

  If users are going to adopt hypermedia systems, then the system must add navigational aids to those which are already provided by the operating environment, in such a way that they are very easy to use. It must be possible to alter the data using the nor mal range of tools available in the system and to add new information, and annotations, for both private and public use. When changes are made to public information, it must be possible to notify users of those changes. There must be a notion of public and private workspaces, and it must be possible to move information (including link information) between these workspaces.

- A system in which all forms of data and media are treated in a conceptually similar manner.

  If the process of making links in one package has a different interface from another package then users will not easily learn to make use of the full range of facilities and media.

3 Defining Open Systems

Much use, and abuse, has been made of the term "open" in computer literature in recent years. Some computer scientists will consider a system open if it may be distributed across a network. Others will say it has an open architecture if there are published details of how to append to, and extend the system. In the field of hypermedia, systems have been referred to as open because the reader was given the same access rights as the author. In defining an open system, we include all of these factors, and more. We believe a minimum definition for an open hypermedia system should be as follows:

1. A system which does not impose any mark-up upon the data which prevents that data being accessible to other processes that do not belong to the system.

2. A system which can integrate with any tool that runs under the host operating system (e.g. MS-Windows 3). Data produced by tools that are not part of the system may be used within the system without adding any special value to that data and without compromising the continued use of the data outside the system.

3. A system in which data and processes may be distributed across a network, and across hardware platforms.

4. A system in which there is no artificial distinction between readers and authors.

5. A system in which it is possible to easily add new functionality. i.e. New program modules may simply be inserted. Clearly a system that conforms to these definitions will go a long way towards meeting the user requirements specified in the previous section.

4 Open Hypermedia Progress With Microcosm

4.1 The Basic Model

Microcosm is currently implemented on MS-Windows 3.1. Versions of the software are also currently under development for the Apple Macintosh and Unix systems running an X-Windows front end. The model was first described in Fountain et al. [2], but has subsequently been fundamentally altered and extended in order to keep all aspects of the system open. Microcosm is best understood as a set of autonomous communicating processes which supplement the facilities provided by the operating system. Microcosm allows the user to navigate through large bodies of multimedia information by a wide range of techniques. No mark-up is imposed on the information, so that all data is accessible to, and editable by, the application that created it. Instead all information concerning links is stored in *link databases* or *linkbases* as we refer to them.

In Microcosm the user interacts with a *viewer.* A viewer is any application in which data may be displayed. Messages to perform actions are sent from the viewer to Microcosm, which then dispatches the message through a chain of *filters.* (See figure 1.) Each of these filters is then given the opportunity to respond to the message by blocking it, passing it on or changing it before passing it on. Based on the message contents, some filters may add new messages to the chain. Eventually the message(s) will emerge from the filter chain and arrive at the Link Dispatcher. This will examine the messages to see if they contain any available actions (such as links to follow), and if so it will offer these actions to the user.

Particularly important filters are the *linkbases*, which contain all the information about link availability. When a message arrives at a linkbase requesting a link to follow, the process looks up the source in a data file and, if found, returns details of the link destination, which are passed on to the link dispatcher. Other filters include processes to aid navigation (such as the History Mechanism) and processes to compute dynamic links.

4.1.1 Links

The Microcosm model allows the user to navigate by a number of different layers of link mechanism.

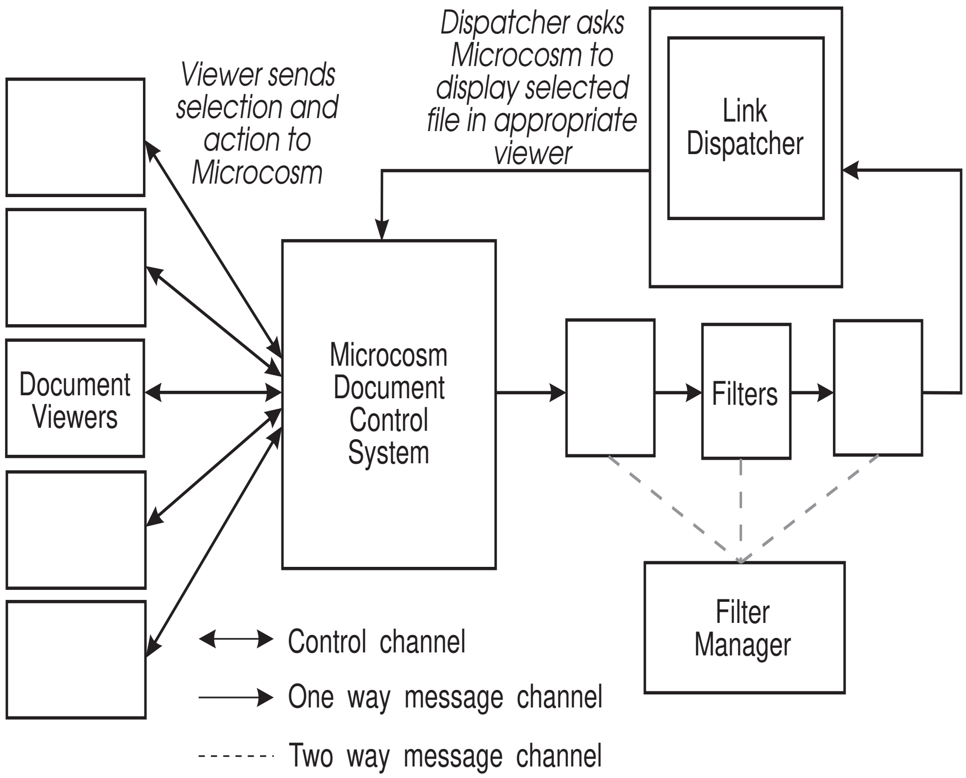

Figure 1: The Microcosm Model.

1. *The specific link* is a link from a particular object at a *specific* point in a source document that connects to a particular object in a destination document.

2. *The local link* is a link from a particular object at *any* point in a *specific* document that connects to a particular object in a destination document.

3. *The generic link* is a link from a particular object at *any* position in *any* document that connects to a particular object in a destination document.

The model therefore allows for *document to document links.* These are links which may be followed from one particular document (or process) to another.

All the above are links with static destinations, in that the destination has been fixed. In all the above cases the destination may be the start of a document, and may also be an active process rather than simply a display of data. The local and generic links, which have dynamic source anchors, are a particularly powerful feature, described more fully in Fountain et al. 1990, which allow a destination to be fixed just once, and subsequently the link may be followed from an appropriate source object in new data or in a new file, as soon as it has been created. Further link levels are:

1. *Text Retrieval links.* These are links which are dynamically computed when requested. There are two ways of achieving such links. The first is to use a *grep* filter, which attempts to match the text selected with the same text in any other document within some pre-defined set of text documents, and returns all possible document names into the link dispatcher. This method is relatively slow. A second method which we have implemented very successfully involves building an inverted index of all the desired documents before the system is used, and using standard information retrieval similarity calculations [8] to match the vocabulary of the selected source with the vocabulary of the documents, and offering the user links to the best matches. This method produces higher quality matches in a much shorter time.

2. *Relevance Links.* If a set of documents have been pre-indexed as required above, then it is also possible to cluster the documents based upon similarity of vocabulary. In such cases it is possible for the user to follow links to other documents in the same cluster.

The mechanisms described above provide a rich and varied set of methods for finding new information, over and above the standard method of navigating the file directory structure, and it should be stressed that the open model makes it possible to add new navigational methods with ease, as and when they are identified and implemented.

4.1.2 Messages and keeping the system open

In order to ensure that the system is kept as flexible as possible, we have adopted a tagged ASCII message format. Any viewer or filter may introduce any tag and data it likes into the message, and any filter will respond to the tags it knows and ignore the rest. Microcosm for Windows makes use of the Dynamic Data Exchange (DDE) facility to pass the messages from process to process. The Macintosh system uses Apple events, and the Unix version uses sockets for the message passing.

Writing a new filter is a simple task that could be undertaken by any Windows programmer without any special understanding of Microcosm internals. A procedure is written to analyse the incoming message for the required tag(s) and to take appropriate actions. This is then inserted into a shell which takes care of all the communication with the DDE channels, and the new filter is then compiled. This flexible modular approach makes it easy for any programmer to make major changes to the functionality of the system.

4.1.3 Viewers

A Microcosm viewer has the task of allowing the user to make selections from the displayed data and then to request an action from a menu of possible actions, such as *follow link, make link* or *compute dynamic link*. This implies that a viewer must be a special program written with Microcosm awareness, and this would appear to conflict with the requirement that any program that displays data may be used as a viewer.

In the general case, this is true. The same problem was confronted in the design of Sun's Link Service [12] , which requires that all applications that wish to use the service are modified to become "link service aware". Pearl argues that if a link service was a standard feature of the operating environment, then all serious applications would be written to make use of this feature. However, in the immediate future this is not likely to be the case and so users of the system are limited to using those applications, such as TextEdit, which have been specifically modified.

We have managed to overcome this problem. In Microcosm we have three types of viewers.

1. *Fully aware Microcosm Viewers.* These are viewers we have written ourselves, which have an action menu as described above. They package up suitable messages and communicate directly with the DDE channel. We have written ten such viewers which deal with data formats that are common in our environment, such as text, bitmaps, video, audio, Windows Meta files and rich text. The advantage of using these viewers is that it is possible to have certain functionality that is not possible in other programs. A Microcosm viewer may display active areas as buttons. These are combinations of object selection and action which are in some way highlighted. Any specific link source may be a button, and the information about what to highlight is stored in the linkbase. The viewer requests this information when loading the data. Microcosm viewers may also be asked to start up with the focus at any particular point in the data.

2. *Partially Aware Viewers.* These are applications from external sources which we have adapted to be Microcosm aware. Many packages such as Word for Windows, Toolbook and Superbase have some level of programmability and access to the DDE. In such applications it is quite straightforward to write the necessary code to produce an action menu and to package a selection and action into an ASCII message for dispatch to the DDE channel. The process of adapting such applications is qualitatively quite different from rewriting the application to become link-aware, and is the sort of task that may be undertaken by any competent user of a package, given appropriate guidance.

3. *Unaware Viewers.* In the worst case, where it is not possible to build into the viewer any form of action menu, and there is no DDE

access, we have introduced the idea of "clipboard links". The user makes a selection from the application in the normal way, and then copies the selection to the clipboard. An action menu may then be chosen from the Microcosm icon, and Microcosm will then take responsibility for taking the clipboard contents and the chosen action and packaging them into a message. A refinement of this approach allows the user to order Microcosm to "monitor the clipboard": whenever the clipboard contents change Microcosm will automatically package the new contents with a pre-selected action, such as *follow-link*.

There are some problems with this approach. The interface to the action menu is not uniform across applications, and since *partially aware* viewers and *unaware* viewers are usually not able to provide information about the exact position in a document at which data was selected, it may not be possible to provide *specific links* from such documents. However *local links* and *generic links* are possible and provide the user with hypermedia functionality from any application.

4.1.4 Filters

All linking functionality in Microcosm is provided by filter processes (see figure 1). As the user interacts with document viewers, messages are generated which are sent to the Filter Management System, or FMS. This process takes care of the propagation of messages to all the currently active filters. The current implementation of the FMS organises the available filters in a simple chain structure as is shown in figure 1. When the FMS receives a message, it passes it to the first filter in the chain, and as messages are returned to the FMS by filters, they are passed on to the next filter in the chain. When a filter receives a message it checks it decides whether it must process the message, and may take action or return the message to the FMS. When messages reach the end of the chain, they are returned to the main Microcosm control process which takes any necessary action.

By implementing the hypermedia functionality of the system as small independent processes in this way, they may be dynamically installed, removed or reordered to vary the behaviour of the system. It is also very easy to enhance the available functions by creating a new filter process, with no need to change existing components of the system. A typical filter is the linkbase, which provides a database of available links. Other examples allow the user to create links or provide navigational facilities. For example the access mechanisms described in section 4.2 use filters to monitor and manipulate system activity.

More detail on the use of filters is given in Hill et al. [6]. Work is currently under way to enhance the functionality of the FMS in order to provide better routing of messages and simplify the user interaction required to change the filter configuration.

4.2 Levels of Access to Information

This open model for hypermedia allows us to provide users with a number of different modes of navigation through information, which are presented here in increasing order of specificity.

4.2.1 Document level browsing

In a recent study of the use of Microcosm [1] we discovered that even when using a heavily linked set of documents with many buttons, users still tended to use the directory structure as a significant method of accessing information. For this reason we have introduced an enhanced file browser which allows the user to store and view attributes for files, in order to aid navigation and document selection. These attributes are in the form of user defined keywords and are accessed by the user as a method of evaluating the usefulness of a particular document when browsing the directory structure or when offered the document as a potential link destination. The Virtual Notebook System [13] uses node attributes as the primary navigational method, but in this case the attributes are selected from a fixed set. In future work we hope to make use of these attributes for computing retrieval links and for automatically creating generic links.

4.2.2 Manually Created Links

These are the specific, local and generic links. Specific links are similar to those provided in most second generation hypermedia systems. Local and generic links require much lower authoring effort and enable us to make links from applications that are not link aware.

4.2.3 Computed Links

These are links which require no manual effort, and range from simple string search techniques, through to full information retrieval techniques. Once a link has been computed it is possible to immediately follow the link or incorporate it permanently in to the linkbase, in which case it is subsequently indistinguishable from a manually created link.

4.2.4 History

In the same way that many operating systems are able to keep a command history by storing a list of all commands issued at the command-line, Microcosm is able to keep a history of all links that have been followed by maintaining a list of all messages sent by the link dispatcher. It is then possible to view these messages in the shape of the file or process that they loaded, and then re-send selected messages.

4.2.5 Mimics

Many current hypermedia applications are in the form of tutorials, and in these cases authors might wish to provide users with fixed pathways through the data space. In Microcosm this facility is provided in the form of Mimics, which form a tour through a set of documents. The user is free to stray from the Mimic, which acts as a optional guide rather than a enforced view as is the case in most other tour systems. A Mimic forms a type of "superlink" as described by Nielsen [10]. The Mimics can be generated in a variety of ways, for example from a user's past history, using a text based language or a form based creation approach.

5 The Advantages Of Open Systems

Adopting the open hypermedia model for Microcosm has enabled us to overcome many of the technological problems faced by other closed hypermedia systems. One of the major problems is that of coping with very large systems (around 10,000 documents or more). This may seem a large number of documents, but even a relatively modest file system will contain this number of files, and we routinely navigate such systems using only the directory structure as a navigation aid. As a result of separating the link service from the files themselves, open hypermedia systems allow the user all of this functionality, plus the added value of link following devices and other navigation aids.

Another advantage of keeping the links separate is that data remains accessible to the original application that created it. Users are free to continue to manipulate data free from artificial constraints created by the system. A further advantage is that it is possible to have more than one linkbase in place at any time. A common configuration for Microcosm is to have one linkbase that contains a set of links over a set of documents that was defined by the original author and for each user to have their own linkbase into which they store their own links and annotations. The idea may be extended to allow shared workspaces. Access permissions to such shared linkbases and the nodes to which they refer are the domain of the operating system, and therefore introduce no new problems for the hypermedia system.

A final advantage of link separation is the facility to process the links. We have produced a program which enables users to manipulate linkbases by such processes as merging links from other linkbases, deleting global references to files that have been removed and changing the scope of links. We envisage further programs that will make use of user defined link attributes.

Authoring effort may be considerably reduced in open hypermedia systems by making use of such facilities as generic links and computed links. These features are not specific to open systems, but the ability to extend the model so that features like computation of links may be added relies upon the system being open.

The system as described applies to any homogeneous file system: that is, any system on which all the files may be accessed and processed by the user's workstation. In the case of Microcosm this implies that all the files are either resident on the workstation or on some network drive. If they are available on a network drive, programs will be loaded onto the workstation before running. However, we see no barrier to allowing the filter processes to run on heterogeneous platforms, and, where the data may be passed between machines in a mutually compatible form, no barrier to allowing nodes to exist on different platforms. Again, the limits to distributing the hypermedia functionality are in the province of the operating system design rather than the hypermedia design. We are currently using an X-Windows based text viewer on a Unix system to use a Microcosm link base and filters running on a remote PC workstation, in order to navigate text files resident on both the Unix system and the PC system.

Finally, by defining an open protocol, open hypermedia systems increase the accessibility of information and will be able to make use of hypertext standards as they mature and become available [4] to allow interaction between different systems. Similarly migration of linked information to different systems will be possible through the use of suitable interchange standards [7].

6 Problems With Open Systems

In common with Sun's Link Service [12] we have found that there are three main difficulties with the open model for hypermedia that we have described. We are currently working to provide solutions to these problems.

6.1 The Size and Position of Anchors

Where a Microcosm aware viewer exists for a particular data format it is always possible to place destination anchors at any point in the data and to follow links to these anchors. However, without adding functionality to applications that are not Microcosm aware, it is not usually possible to start up an application at a specific point, such as at a given paragraph in a text document, or a specific item in a CAD drawing. The situation is even worse if the destination is a particular state of a dynamic application such as a particular frame in an animation. The best we can currently do is to is to start the application with the chosen data set, and then leave it to the user to navigate within the application to the required point.

In the majority of applications we have found this approach to be perfectly adequate. Where hypermedia is being used as an extension to the directory system as a method of navigating a large information space, this level of granularity in discovering the correct destination is all that users expect, and they are quite happy to continue from the start of a document or data set by browsing with whatever tools are provided by the application itself. Where authors have produced a more finely linked and usually more static hypermedia product, such as a tutorial, they have almost invariably used the Microcosm aware viewers which provide the ability to link to specific points.

Various possibilities exist to improve upon this situation. The most easily available is the use of system macros. The creator of a link into an OEM package may produce a macro that moves the focus to the item of interest in the data. When the link is followed, the package is started with the appropriate data set and then the macro is replayed.

A further problem with maintaining the linkbase separately from the document is that it is not immediately clear what links are available to the user. In closed systems there is usually some form of visual clue, such as bold text, which indicates that a particular object is "live" and may be clicked on. Microcosm allows authors to make specific links into buttons, and link aware viewers will query the linkbases for any buttons and display the objects in some visible way. However, it is not feasible to check for all generic links in this way. We have investigated two solutions to this problem.

One solution involves holding in memory an index to the complete set of links, and on being asked to *find all* links, the viewer attempts to match the contents of the current window against the index, and displays all links it finds in the current window, The second solution involves the user in selecting an area of text and asking to *show links*. The selected text is split into words and word pairs and repackaged as follow link messages. The links found will appear in the available links box. The first solution is more user friendly but may only be used from Microcosm aware text viewers whereas the second method has the advantage that it may be used from *any* text viewer.

6.2 The Editing Problem

Since links are separated from the nodes, there is a problem if documents are edited. In Microcosm, link source and destination anchors are recorded in terms of absolute position within a document. If objects are inserted or deleted before a link anchor, then the information in the linkbase will become outdated. This is not a problem if links are "document to document" rather than links where one or other end is fixed somewhere specific in a document.

The situation gets worse if a file which is the destination of a link is moved or deleted. Then the link will dangle. This is particularly a problem in a distributed file system where file availability may depend on network access or where links may point to files in users' private workspaces.

6.3 The Versioning Problem

It would be tempting to dismiss the problem of version control as another matter that is entirely within the domain of the operating system. After all, Unix provides version control systems such as SCCS, and many good implementations of RCS are available for PC's. Indeed, if all that was required was to keep old versions of documents as updates were made, then the facilities provided by the operating system would be quite adequate. Unfortunately, hypermedia presents new problems. If documents have been edited, then are the links to the old version or the new version? Do we need to keep versions of the linkbase as well? In open hypermedia systems the versioning problem becomes intertwined with the editing problem.

6.4 Looking for Solutions

The current solution that we use in Microcosm is as follows. All links contain a date stamp for the source and destination documents, which should match with the document's own date, as held by the operating system. In order to provide some idea of workspaces the document management system (DMS) keeps attributes which state which application or applications a particular document is us within. Another document attribute is an alias for the document's name. This alias is a better description of the document than can be provided by a small filename, and also makes it possible to move the file, updating only the alias in the *DMS*, rather than all the references to the document in linkbases, some of which might not even be online. This solution helps to minimise problems that may occur due to early structural organisation of a system turning out to be inappropriate at a later stage.

- Whenever a document is loaded all *specific* links from the document are checked. If any of these links have a date for the source document that is earlier than the date of the document itself, the user is warned.
- Whenever a link is followed to a document, the date of the destination document is checked against the date of the link. Again, if they do not match the user is warned, or if the destination document no longer exists, the user is warned.
- A link aware editor for text documents exists, which when loaded finds all references to the document in linkbases that are currently available, and updates these link anchors when editing is complete. During the edit the document is locked against other use.

This solution keeps the user aware of any inconsistencies, but makes no attempt to fix them when they occur. Clearly it would be best to attempt to ensure that such inconsistencies do not occur so we advise authors to try to avoid the use of specific links where possible. Document to document links and generic/local to document links have far less scope for becoming inconsistent. Also we advise that all linkbases used with a specific application should be kept on-line so that the link aware editor may update them.

As has been explained earlier, the task of keeping an open hypermedia consistent is non-trivial. There are a number of preconditions under which such a system will remain consistent and at present we are attempting to create such a system. These preconditions are:

1. No edits may be made to documents of which the link service is unaware.
2. The document management system is aware of all linkbases that might be effected by any edits, and has access to these linkbases in order to make suitable changes.
3. There is an algorithm for moving specific link anchors from an old version of a document to a new version.
4. Versions of documents and link bases are maintained automatically, and on traversing a link to a document, the user is able to step back through previous versions of the document, but the most recent version of a document must always be available outside of the versioning system for access and processing by external applications.

7 Conclusions

This paper has presented a view of hypermedia as an operating system service for integrating information and processes in a distributed heterogeneous environment, and allowing users rich and varied access methods to navigate through that information. The system described is able to perform all the tasks that might be expected of a simple second generation hypermedia system, although there are some speed penalties to be paid for the cost of keeping the system open. However, open systems introduce new problems concerned with binding link anchors to the data so that the system remains consistent. Two solutions have been discussed in this paper, but this is clearly an area in which further work is required, particularly as the size of such networks of information increases. Furthermore, some of the techniques that have been investigated in this paper refer only to documents that contain text. It is not yet clear how these techniques may be extended to deal with generalised multimedia, nor indeed whether it is appropriate to attempt to do so.

True open hypermedia systems are a relatively new concept. In the long term it seems likely that operating system vendors will provide increasingly useful link and object management systems as an integral part of the operating system and application software will be written so that it is aware of these services.

References

[1] Hugh Davis, Wendy Hall, Gerard Hutchings, David Rush and Rob Wilkins, Hypermedia and the Teaching of Computer Science: Evaluating an Open System, in: David Bateman and Tim Hopkins, eds., *Proceedings of the First Conference on Developments in the Teaching of Computer Science*, University of Kent, 1992

[2] Andrew M. Fountain, Wendy Hall, Ian Heath and Hugh C. Davis, MICROCOSM: An Open Model for Hypermedia With Dynamic Linking, in: A.Rizk, N.Streitz and J. Andrè, eds., *Hypertext: Concepts, Systems and Applications. The Proceedings of The European Conference on Hypertext*, Cambridge University Press, 1990

[3] Halasz, Frank G., Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia systems, *Communications of the ACM*, 31(7), 836–855, 1988

[4] Halasz F. G., Schwartz M., The Dexter Hypertext Reference Model. In *Proceedings of the NIST Hypertext Standardisation Workshop*, NIST Special Publication SP500-178. 1990

[5] Halasz, Frank G. "Seven Issues": Revisited, Hypertext '91 Keynote Talk, 1991

[6] Gary Hill, Rob Wilkins and Wendy Hall, Open and Reconfigurable Hypermedia Systems: A Filter Based Model. *Computer Science Technical Report 92–12*, University of Southampton, 1992

[7] Leggett J. and Killough R. L., Issues in Hypertext Exchange. *Hypermedia* 3(3), 159 – 186, 1991

[8] Zhuoxun Li, Wendy Hall and Hugh Davis, Hypermedia Links and Information Retrieval, *The Proceedings of the 14th British Computer Society Research Colloqium on Information Retrieval*, Lancaster University, 1992

[9] Kathryn C. Malcolm and Steven E. Poltrock and Douglas Schuler, Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise, *Hypertext '91 Proceedings*, ACM Press, 1991

[10] Nielsen J., *Hypertext and Hypermedia.* Academic Press, London, 1990

[11] Murugappan Palaniappan, Nicole Yankelovich and Mark Sawtelle, Linking Active Anchors: A stage in the Evolution of Hypermedia, Hypermedia, vol 2, num 1, 1990

[12] Amy Pearl, Suns's Link Service: A Protocol for Open Linking, *Hypertext '89 Proceedings*, ACM Press, 1989

[13] Frank M. Shipman III, R. Jesse Chaney and G. Anthony Gorry, Disributed Hypertext for Collaborative Research: The Virtual Notebook System, *Hypertext '89 Proceedings*, ACM Press, 1989

[14] Nicole Yankelovich and Bernard J. Haan and Norman K. Meyrowitz and Steven M. Drucker, Intermedia: The Concept and the Construction of a Seamless Information Environment, *IEEE Computer*, Jan 1988, 81–96