Light Hypermedia Link Services: A Study of Third Party Application Integration

Hugh C. Davis, Simon Knight and Wendy Hall
The Image and Multimedia Research Group
The Department of Electronics and Computer Science
The University of Southampton
Southampton, UK.
SO17 1BJ
fax : 44 703 592865
e-mail: {hcd,sjk92r,wh}@ecs.soton.ac.uk

ABSTRACT

Recently there has been a tendency for the research community to move away from closed hypermedia systems, towards open hypermedia link services which allow third parties to produce applications so that they are hypertext-enabled. This paper explores the frontiers of this trend by examining the minimum responsibility of an application to co-operate with the underlying link service, and, in the limiting case where the application has not been enabled in any way, it explores the properties and qualities of hypermedia systems that can be produced. A tool, the Universal Viewer, which allows the Microcosm Hypermedia System to co-operate with applications which have not been enabled is introduced and a case study is presented which demonstrates the functionality that may be achieved using entirely third party applications, most of which have not been enabled.

KEYWORDS: Open Hypermedia, Hypermedia Link Services, Integration, Microcosm.

1 INTRODUCTION

The introduction of HyperCard led to a multitude of hypertext-like applications in the late eighties: keynote speakers at conferences confidently predicted that hypertext, and, shortly afterwards, hypermedia, were good ideas whose time had come. However, the take-up of the technology has not advanced at the expected speed: the hypermedia community has attempted to explain this by suggesting that systems are functionally poor: while the currently available commercial systems have reasonable user interfaces, they may not offer sufficiently rich link types, composite or virtual structures, links to information retrieval, intelligent dynamic link creation, methods for coping with temporal media etc.. While all this research has been useful, there has been no great rush to introduce new generation systems to the market, and speakers from industry [6,13] have been suggesting that in order for hypertext to realise its potential, it must support and integrate with the tools in the desk-top environment.This is in contrast with the "insular, monolithic packages that demand the user disown his or her present computing environment to use the functions of hypertext" as described by Meyrowitz [14].

Perhaps surprisingly the requirement for integrating hypertext with desktop applications was identified as long ago as 1978 when Engelbart described the ability to interface with other systems and applications as one of the most important features of Augment [4]. More recently Malcolm et al. [13] suggest that industry requirements include the need to produce an adaptable environment for the integration of tools and services, which is distributed and platform independent, and which enables users to easily find, update and exchange information.

Greif [6] describes the benefits gained if applications are hypertext enabled, and sets the challenge to the research and development community, to produce lightweight systems that support applications. This paper addresses this challenge, and investigates the extent to which these goals may be realised using an open hypermedia link service and current desktop applications. We describe work that has been undertaken by the Image and Multimedia Research Group at the University of Southampton, UK, to integrate link services with the applications commonly found on the desktop: in particular we describe work that explores the functionality that may be achieved in the limiting case where the application cannot be adapted in any way to co-operate with the link services. We refer to the resulting system as a light hypermedia link service.

2 BACKGROUND

Any system that is intended to integrate applications will need to be an open hypertext system [3,9,11,15,16,18], in the sense that it will need to be extensible, tailorable and possess some published interface whereby content viewers may communicate with the link service. For example in Multicard [16] this is achieved using scripts and in Microcosm it is achieved using a tagged message format.

We have identified a number of levels at which system builders have attempted to integrate applications as data viewers for hypertext systems. In increasing order of generality they are:

1. *Tailor Made Viewers.* These applications are written specifically for integration with the hypertext system.

2. *Source Code Adaptation.* Where the source code for an application is available it is possible to add the code for communicating with the hypertext link service.

3. *Object Oriented Re-Use.* A basic hypertext viewer class is created and viewers for specific data types inherit from this class. The KHS [2] mail viewer is an example of such a system.

4. *Application Interface Level Adaptation.* Many packages provide flexible interfaces and macro programming languages, via which it is possible to add Hypertext functionality.

5. *Shim or Proxy Programs.* These are programs that sit between the hypertext link service and the viewer, translating actions in one system to actions that the other can understand. The Microcosm Universal Viewer described later in this paper is such an example,

6. *Launch Only Viewers.* This is the worst case, when all the hypertext system can do is to launch a given program with a given data set, but from the program there will be no hypertext functionality.

One of the factors that has made it so difficult to integrate existing hypertext systems with desk-top applications has been the adoption of the Dexter reference model [7] which requires that the content viewer / editor is responsible for handling link anchor identifiers. Adapting existing applications to do this involves considerable enhancements which will normally only be achieved by adding code at the source code level [15,18] although it may be possible to add this functionality using a macro language where this is sufficiently powerful (e.g. EMACS has a lisp like programming language with which one can actually tailor the environment [16,18]). Users are unlikely to wish to invest the effort to tailor their applications, and software houses are not likely to wish to add link anchor handling facilities until some standard emerges for doing so.

An alternative approach to embedding anchors within the node component is to require the link to hold the information about the anchor position, and the link will be held externally in a link database. This is the approach taken by the Microcosm Link Service which has been reported elsewhere [3,5]. The SB3 system [11] takes an intermediate approach in that the link information itself is separated from the component information, but anchors refer to persistent selections which are in effect anchor *positions*, which must still be maintained by the content viewer process.

The following section describes the important aspects of the Microcosm Link Service so that the user may follow how this architecture assists in creating light hypertext services.

3 THE MICROCOSM LINK SERVICE.

The Microcosm system assumes that the node content viewer is able to display an action menu (containing such items as *Follow Link, Compute Link, Start Link* and *End Link*) and is able to communicate the chosen action, along with the details of the selected object, the position of the selected object and the file in which the selected object was found, to Microcosm. This process then takes care of passing this message through a set a programs known as *filters,* which all have the opportunity to react to the message, as shown in Figure 1.

Three important filters are:

The Linkbase Filters
These filters communicate with database files which storethe links (linkbases). When they have a link which matches the source domain in the *Follow Link* message, they respond by creating another message asking Microcosm to dispatch the appropriate content viewer so that the user may view the end of the link.

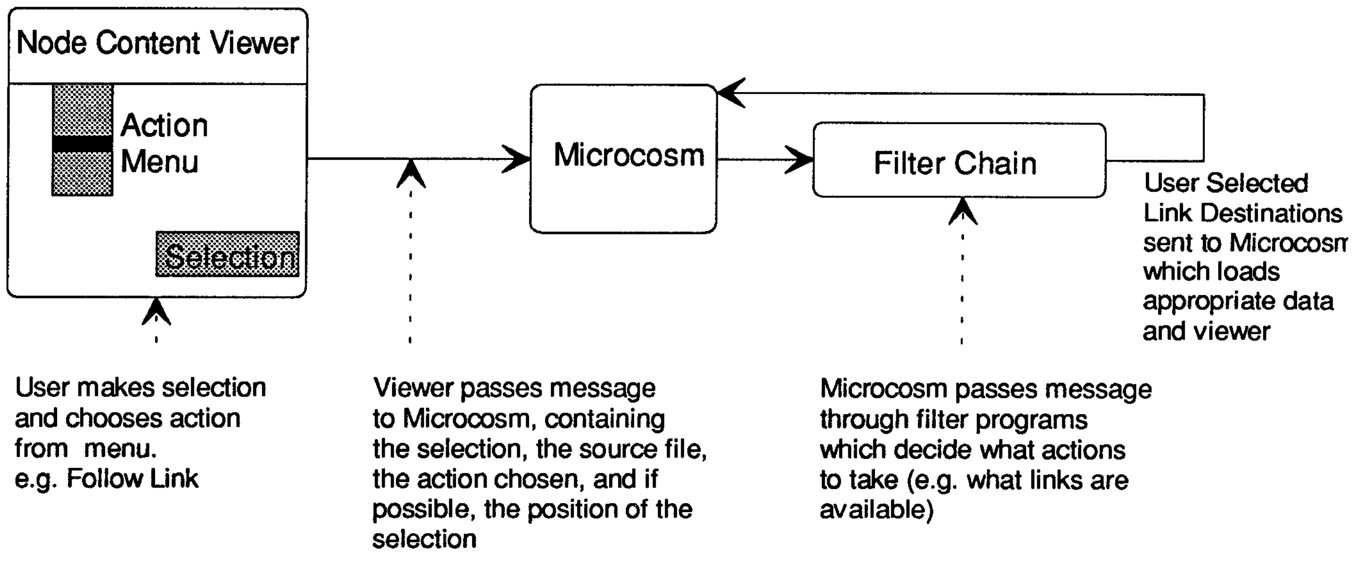

Node Content Viewer
Action Menu
Selection
Microcosm
Filter Chain
User Selected Link Destinations sent to Microcosm which loads appropriate data and viewer
User makes selection and chooses action from menu. e.g. Follow Link
Viewer passes message to Microcosm, containing the selection, the source file, the action chosen, and if possible, the position of the selection
Microcosm passes message through filter programs which decide what actions to take (e.g. what links are available)

Figure 1: Message Passing in Microcosm.

The Linker
This handles messages to create a link (starting at some source selection) and to terminate it (at some destination point or selection). The information about the source and destination is stored in a linkbase file.

The Computed Linker
This uses a pre-created index of some set of text files to apply information retrieval techniques [12,17] to attempt to provide links where none have been manually created.

The format of the linkbase files is of particular importance, because it is this format that allows us to interact with third party applications. A linkbase consists of an indexed text file. Each link is represented by a set of tagged fields. e.g. The set of tags in Figure 2 describes a link form the text "SPC" in a text file (which the system knows by a unique document identifier) and occurs at 214 characters through the file.

```
\SourceFile 100.02.24.93.11.39.54
\SourceSelection SPC
\SourceOffset 214
\SourceDocType TEXT
\DestFile 100.02.24.93.12.40.49
\DestSelection manual
\DestOffset 312
\DestDocType TEXT
\Description SPC Definition
```

Figure 2: Example Tagged Link Description

If a message were sent to the linkbase, asking to *Follow Link* from this source document, containing the selection "SPC" chosen at the offset of 214 characters through the file, then the destination document would be dispatched, with the cursor at the line containing the 312th character (the start of the word "manual"). We call this a *specific link.* In Microcosm a button is a specific link that is coloured by the viewer so that it may be simply clicked on to follow the link.

The absence of a SourceOffset in the linkbase would indicate that this link could be followed wherever the text SPC was chosen in the source file. We call this a *local link.*

The absence of both a SourceOffset and a SourceFile in the linkbase would indicate that the link could be followed wherever the text SPC had been selected. We call this a *generic link.*

Note that offsets here are described in terms of bytes through the file, but this is not pre-requisite. The semantics of the offset are decided by the viewer and can be resolved by the linkbase so long as the method for expressing the offset is used consistently by the viewer for this data type.

It is important to note that, except in the case where there is a button indicating the presence of a link anchor, it is normal for users to make selections themselves, and then request an action such as *follow-link.* It is of course possible to select fairly large areas of interest, such as a paragraph of text, and the system will then find all anchors and corresponding links available within that selection.

4 VIEWER COMMUNICATION PROTOCOLS

A fully aware viewer is one that can participate in all the Microcosm communication protocols. However it is possible to participate in only a subset of the complete set of protocols. This section informally describes the Microcosm protocols and examines the consequences of ignoring them.

1. Launch document
It must be possible to execute a viewer program with a given data set loaded from external code. This protocol is prerequisite in order to provide destinations to hypertext links.

2. Display Buttons
A fully aware viewer will, as soon as it starts, send a message back to Microcosm asking for details of any buttons (specific links that are to be highlighted in some way): Microcosm will respond by sending back the button information which the viewer will use to repaint the screen. However, if the viewer does not request buttons, the information will not be sent, so buttons will not be painted. This does not mean that the links are unavailable, but simply that the user is made responsible for selecting the source anchor and choosing *Follow Link* from the action menu. This means we might have hypertext nodes without buttons, a viewpoint that might please some [8], but others might feel that it is at times necessary to indicate to the user what options they might have to continue browsing.

3. Start-Up Options
When a viewer has started-up it may be desirable to indicate the destination anchor in some way. In the case of a text document this might mean scrolling the document so that the anchor text is on the top line: in a video this will mean moving to the correct frame: in a drawing this might mean highlighting the relevant object. Alternatively, it is possible to do away with destination anchors altogether and to simply load the given document which will mean that the resulting hypertext is very "chunky", leading to a notecard style hypertext, where the destination of any link should preferably be limited to some sensible size.

Also, when a viewer has started-up, it is possible for the viewer to communicate with Microcosm asking for any display details, such as font sizes, window sizes, window positions, background colours, button colours etc. Although these features are all useful, none of them are essential, and have mostly been implemented to overcome authors' concerns about the use of the vanilla Windows interface for delivering materials to naïve users.

4. Check Link Integrity
When a viewer displays a document it is desirable to check that any offsets within the document used to describe link anchor positions still correspond to the required objects. This may be achieved by requesting that the linkbases provide a list of all links containing specific source and destination anchors within the current document. If these links are dated after the date that the document was last edited then they are safe, but if the document has been edited since any link was created, then the viewer must relocate the anchor (if it can) and update the linkbase.

If the viewer is incapable of entering into this dialogue with the linkbases, then it is possible that anchors might be incorrectly located. Possible solutions to this problem are:

- Make the document read only, so that it may not be edited. If it must be edited, produce another version.
- Do not use specific link anchors. All links should be generic or local, and have their destinations at the top of, rather than within, the destination file.
- Use a search engine to find link anchors rather than offset / position information. This is the approach taken by HyperTed [19]: then as long as the anchor is unique within the file, and the anchor itself is not changed, it will be possible to edit the file.

5. Service User Actions
Once users begin to interact with the system they will make selections and choose actions from the menu (or click on buttons where they are available). The viewer is responsible for

- providing an action menu
- identifying the selection made
- identifying (in any way appropriate to the application) the position of the selection
- identifying the current data file
- packaging all of the above into a Microcosm message and sending it to Microcosm.

If the application is unable to identify the position of the selection, then Microcosm will only be able to provide local and generic links: specific links would be impossible.

5 ENABLING APPLICATIONS FOR HYPERMEDIA USE

Many serious applications these days have a built in macro programming language, and some degree of adaptability so that new menu options may be added. For example Microsoft Office applications have Visual Basic for Applications, the Lotus Office Suite applications have macro languages and AutoCAD has AutoLisp. Using these facilities it is very simple to add the standard Microcosm menu to such an application, then to add appropriate macros for each action chosen. Figure 3 shows a Word Basic macro for the *Follow Link* action, which uses Microsoft Windows' DDE to communicate with Microcosm.

In this case no attempt has been made to send any source anchor position (zero was sent) which means that Microcosm will only attempt to follow generic and local links. Word may be persuaded to yield position information (in terms of words through the file) but in this case the macros become larger.

```
Sub MAIN
    On Error Resume Next
    If Selection$() = "" Then
      MsgBox("No Selection Defined!")
    Else
      chan = DDEInitiate

      ("MICRCOSM","LinkServer")
      If chan = 0 Then
          MsgBox("Could not connect
                        to Microcosm")
      Else
          DDEExecute(chan,
            "[FOLLOW.LINK]
            [" + Selection$() + "]
            [" + FileName$(0) + "]
            [0][DOC]
            ")
          DDETerminate(chan)
      End If
End Sub
```

Figure 3: Word Basic Macro to Follow Link

Other packages may be similarly adapted, for example spreadsheets such as Excel may use a range name to describe anchor positions and AutoCAD allows the author to associate a unique name with each object: this name may be used as the anchor identifier for linking purposes.

An example of such a hypertext enabled application is shown in figure 4.

Packages with macro languages may also be programmed to launch macros at start-up which alter the current view so that the destination anchor is clear (e.g. by highlighting or scrolling to the appropriate line of text.)

Some of the applications we have enabled using this method are Word for Windows, Word Perfect for Windows, AMI Pro, Lotus 123, Excel, Toolbook, Authorware Pro, AutoCAD, Microsoft Access and Microsoft Project.

The Chimera System [1] takes a similar approach to Hypertext enabling third party applications. In the case of FrameMaker, which publishes a set of RPC calls, a proxy or "wrapper" program handles communication between Chimera and FrameMaker, and FrameMaker macros have been used to observe user actions and to communicate them to the wrapper. Multicard [16] has also used macros within editors such as Emacs for hypertext enabling.

6 APPLICATIONS WHICH CANNOT BE ENABLED: THE UNIVERSAL VIEWER

Perhaps the most interesting result of this research has been the realisation of the power of the hypermedia functionality that it is possible to achieve with applications that have no communication facilities whatsoever other than those that are standard within the host operating environment.

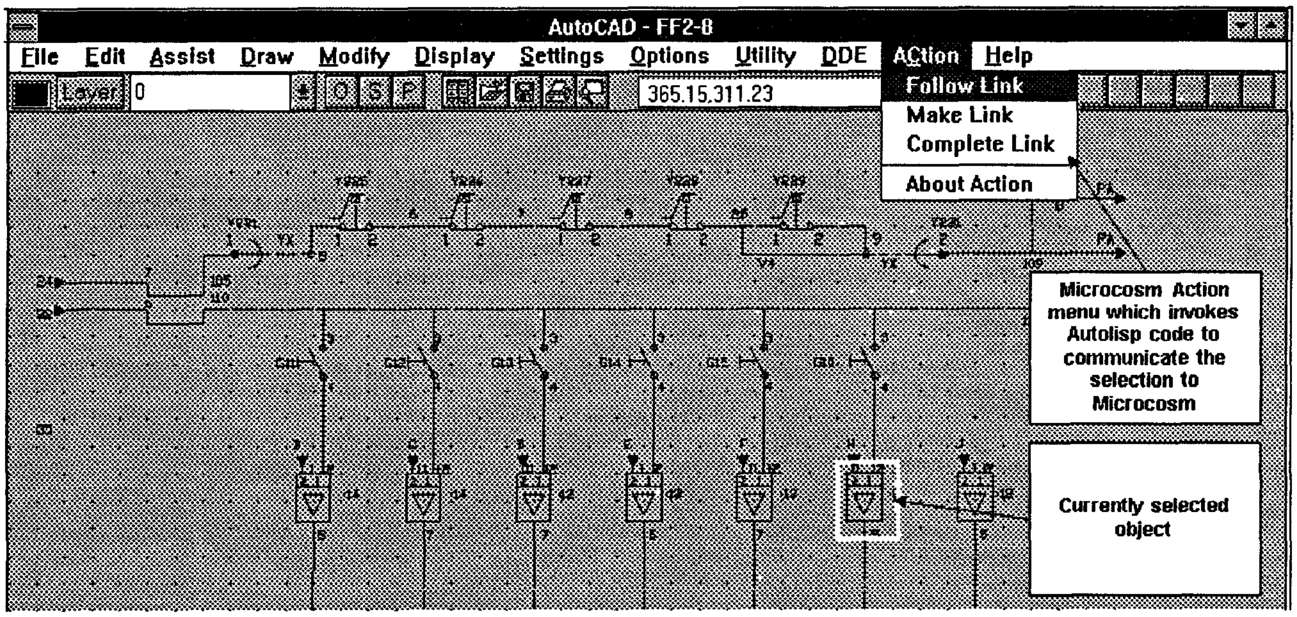


Figure 4: AutoCAD (adapted as a semi-aware viewer) following a link

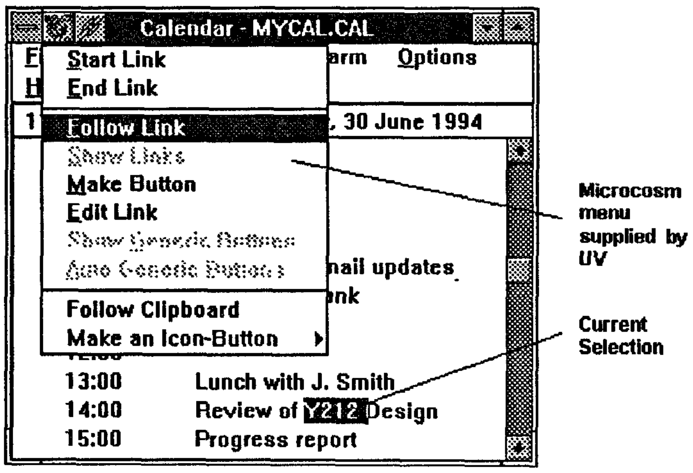


Figure 5: Microsoft's Calendar running under UV. The user is about to follow a link.

Earlier versions of Microcosm allowed the user to set an option within Microcosm that told Microcosm what action to perform on any new data entering the clipboard. Thus the user could run *any* program that supported the clipboard, and every time some data was copied from the application, Microcosm would take the appropriate action, such as Follow Link or Compute Link would be performed. This approach seemed very powerful, but the interface to this functionality was so obscure that it was rarely used. (Selecting Copy from the Edit menu is not an instinctive interface to link following!)

In order to improve the interface and to further investigate this branch of the research we designed a "parasitical" program that we called the "Universal Viewer" or UV. A list of applications with which we wish UV to co-operate are maintained, and UV monitors Microcosm events in order to identify when a program on this list is launched: when it identifies such an event it attaches a small Microcosm icon to the application's title bar (e.g. see figure 5). Clicking on this icon produces a menu, identical to the action menu on any other viewer. The UV is a "shim" which is responsible for all communication with Microcosm, and handling any communications that are possible with the application.

A similar approach has been taken by the designers of the ABC system [10] who were working with the X-Windows system. They re-parent the Window of the required applications so that the new parent is the Generic Function Manager (GFM). To the user this appears as if the application is running with an extra menu bar inserted between the original application title bar and the application work space. The GFM allows the user to make node to node hyperlinks, but they have not attempted to handle anchors of any kind, unless the application is adapted at the source code level.

In order to co-operate with an application, UV needs to be pre-configured to know a few details.

- When Follow Link (or some other action) is selected from the UV menu, how will UV communicate to the application that the application should pick up the current selection?

  There are a number of options. The most common is to use a UVMACRO which is a list of menu selections, (conceptually similar to a set of keystrokes) which will cause the application to pick up the current selection. Other options are to use DDE (where it is supported by the application), to use a Windows Recorder File (keystrokes again) or NONE, when this is not possible or meaningful as the application has no concept of selections.

- When the selection has been made, how will the application communicate that selection to UV?

  Options are via the clipboard, via DDE, via a named swap file or, again, NONE where this is not possible or meaningful.

- What options will be available for locating a destination anchor when a link is followed to a document in this application?

  The problem here is to change the unaware application into a state where it displays the destination anchor, e.g. by scrolling a text document until the anchor is within view, and then highlighting the text span. To achieve this links may be created which launch a named keystroke macro when the destination file is loaded. These named macros must be pre-defined for each application. A typical such macro is FINDSELECTION, which will take the text of the destination anchor (from the link information) and run the application's search engine to find the first occurrence of that text. The list of available macros are offered to the user at the time that a link is created, as shown in figure 7.

In the case where it is not practical or possible to drive the application into some appropriate state to display the destination anchor, then the only option will be to launch the application in its normal start up state. An option which slightly improves upon this worst case is to launch the application and at the same time display a USERNOTE which will be a message to the user instructing them how to proceed so that the meaning of the destination of the link is clear.

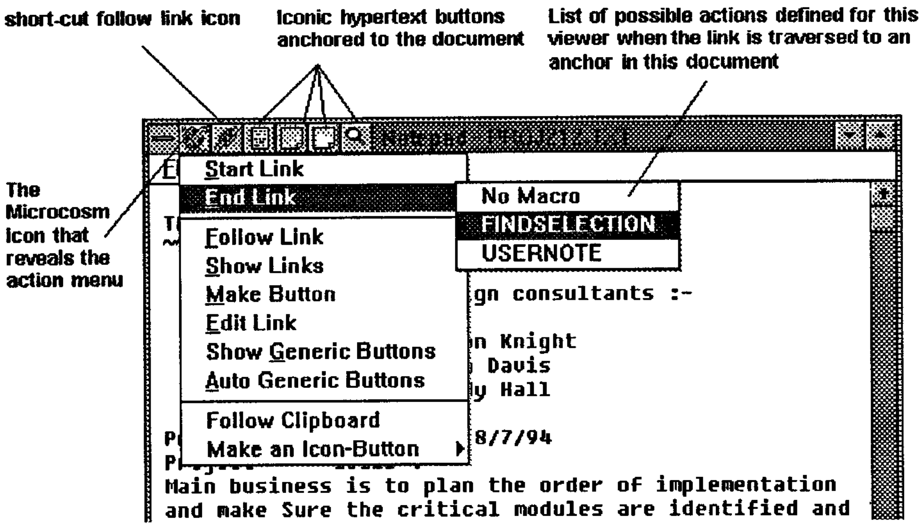
The resulting system gives us local and generic links out of many applications, and gives us methods of following links into specific points within application data: however buttons are still missing. In order to generate buttons we introduced the idea of adding small user definable graphic icons onto the title bar. These represent buttons that may be clicked at any time. When the cursor is moved over such a button, text appears describing the link, and putting the link into context, since its position on the title bar gives no clue as to the circumstances under which one might wish to follow this link. Although this facility is limited when compared to highlighting buttons in context, it is still surprisingly effective.

7 CASE STUDY

Much of this work was motivated by the paper "Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise" [13]. We decided to build a small hypermedia application using entirely third party applications as data viewers, in order to address some of the issues identified. With due respect to the Boeing team,

Fully Aware Viewers	Applications running with Universal Viewer
When loaded as result of a link following operation, the viewer loads the data so that the destination anchor is highlighted in some way.	When loaded as result of a link following operation, either the data is shown at the start, or a stored keyboard macro executes to display the destination anchor.
Viewer asks for buttons relevant to this data, which are then highlighted in context.	Universal Viewer asks for buttons relevant to this data, which are located on the title bar of the application.
Author may pre-set viewer display parameters such as window size and position, fonts, colours etc.	Window size and position may be pre-set but other parameters are determined by operating system defaults.
Typical actions on the action menu will be, Follow Link, Compute Link, Show Links, Start Link and End Link. These menus are configured dynamically by querying which filters are currently available.	Actions on the action menu will be the same as the fully aware viewer. However it will not be possible to author *specific* source links, and destination anchors are resolved by content search rather than position.
If the document is edited the viewer will need to be able to communicate with the linkbases to resolve any links that have been moved.	Since there are no specific source or destination anchors, the document may be freely edited. The only problem that might occur is if a feature used, e.g. by a search engine, to locate the end of a link was altered, removed or duplicated.
Viewers have been written to produce links into and out of temporal media (digital sound and video).	No facilities for dealing with temporal data, other than launching the data at the start point, and following links from the whole document.
A show links action on the menu will find all links within a given selection and either highlight them within the current viewer or list them in the available links box.	A show link action on the menu will find all links within the current selection, and either display them as graphic icons on the title bar, or list them in the available links box.

Figure 6: Contrasting the facilities provided by the fully aware viewers with those provided by applications running with the Universal Viewer.


short-cut follow link icon
Iconic hypertext buttons anchored to the document
List of possible actions defined for this viewer when the link is traversed to an anchor in this document
The Microcosm Icon that reveals the action menu
Start Link
End Link
Follow Link
Show Links
Make Button
Edit Link
Show Generic Buttons
Auto Generic Buttons
Follow Clipboard
Make an Icon-Button
No Macro
FINDSELECTION
USERNOTE

```
gn consultants :-

n Knight
 Davis
y Hall

8/7/94

Main business is to plan the order of implementation
and make Sure the critical modules are identified and
```

Figure 7: The project Notebook. The User is about to create a destination anchor in this document. In this case the anchor will a particular text string.

we borrowed their idea of a "scenario" with which to describe the functionality of our application.

Jane is a development engineer working for a cable manufacturing company. On arrival at work each day she consults her calendar, and today she sees that she has scheduled time this morning to chase progress on a project to replace the drive system on the area production line that winds the finished cable on to drums prior to dispatch. (e.g. see figure 5).

She follows a generic link from the name of the project in her calendar to the project notebook, which contains details of those responsible for aspects of the project, the project goals, and a list of technical issues. She sees graphic icons indicating that there are links to other documents, including engineering drawings, circuit diagrams and graphics detailing all aspects of the technical and commercial management of this project.

Having reminded herself of the goals of the project, Jane follows a link from the project notebook document to the programmer's PERT chart held in MS-Project where she discovers that the project is currently being delayed by the lack of a final approved design for the sub-system that will provide the speed demand to the motor-drive. She follows a link from the name of this sub-system to the minutes of the monthly design review meeting, where she discovers that while the original design was completed on time, it had failed on test to meet the dynamic requirements, detailed in the specification document.

Following a link from the project notebook to her e-mail folder on this project, Jane browses her mail till she recalls that she had been informed of this problem, and that a discussion group had used Lotus Notes to brainstorm the required modifications. Following a link to this information, Jane discovers the rationale behind the new design before contacting Peter to find out the latest state of the project. Peter informs her that the re-design is now complete but that all the drawings and links to them are still in his private workspace. He moves this information into the public workspace so that Jane may inspect them. Jane discovers that a circuit design, held in AutoCAD is available. Links are available from this drawing to a circuit simulation program which runs on a remote machine and to the results of the tests validating the dynamic behaviour of the new sub-system. Viewing the revised circuit diagram she finds that a component has been used of which she is unaware: Jane knows that generic links are made from all electronic component names to their entries in a database holding details of the company's preferred components. She follows such a link by clicking on the component in the AutoCAD diagram and discovers the full details of the component, including its characteristics and cost. By reviewing this information she confirms that there is little risk in using this new component in the final design.

Jane pulls up a spreadsheet, and by following links on all the major circuit components in the diagram to the component database, creates a spreadsheet estimating the total cost of the new sub-system, then following a link from the project notebook to the project costs spreadsheet, she pastes in the new cost of this sub-system. Changing this *spreadsheet automatically causes the system to send e-mail to the interest group informing them of the change.*

Jane spends the next few minutes making a few additional comments in the project notebook, and adds links to the new AutoCAD drawing. She is now confident that she can authorise the release of the design for manufacture. In doing this the status of the drawing is changed, preventing any further unauthorised modification.

She then adds a few private annotations regarding activities undertaken during the final test, the drawing and links to them. On completion of the final task she returns to her calendar to see what she must do next. As she leaves this activity a record of the session is stored: this will prove valuable at the time when a full quality audit of the design has to be undertaken prior to final acceptance.

This scenario, although shorter, is functionally similar to the Boeing scenario: the hypertext is "chunky" in the sense that the links are mainly node-to-node or using generic and local links, but the application is surprisingly effective.

8 CONCLUSIONS

In this paper we have discussed the design issues that must be considered in order to provide hypertext functionality to third party applications, either with a minimum of adaptation using application macro facilities, or with no adaptation.

The key to success in this endeavour is to separate both the link and the anchor information from the node content, so that the application is not required to provide the functionality to handle anchor identifiers within the data content. Instead, all that the application must be able to provide is the selected object (such as the selected text string), the name of the current data file and the action chosen by the user. In modern GUI operating systems, the operating system itself is usually able to provide of all these details except the action chosen by the user, and to this end we have provided a shim program known as the Universal Viewer, which can provide this information. Separating the anchors from the data has side effects with regard to the integrity of links when editing the data. We have discussed these problems and a range of solutions.

The resulting hypertext system differs from its predecessors in two significant aspects. Firstly, navigation is primarily node-to-node or local/generic anchor-to-node. Secondly, link anchors are generally not displayed as buttons or highlighted objects in the way that users have come to expect This issue is discussed in the paper "Ending the Tyranny of the Button" [8], which urges that users should be encouraged to expect to query the system for links in much the same way as they might query an encyclopaedia for information: in any system where links may be generated dynamically every object in the document potentially becomes the source of a link; having the whole document highlighted would be intrinsically as useful as having no buttons at all. In Microcosm the interface to hypertext functionality is achieved by taking a selection and identifying if the selection, or any part of the selection, represents a resolvable anchor. This makes it possible for users to ask what links are available within, for example, a whole paragraph of text.

It is clear to the authors that in the future much of the functionality that we have described in this paper should be, and perhaps will be, implemented within the operating system. If the operating system allowed users to attach extra menus to any application (in much the same way as Microsoft use Visual Basic for Applications) and would also provide functions to return information about the current selection, such as the name of the selected object or the position of the selection, then link services could provide a higher degree of specificity for link anchors. The Apple Events suite for inter-application communication makes a start at providing these sort of facilities. The editing problem would be simplified if applications, such as the link service, could register in an "interest set" that would be notified when certain actions occurred, such as files moving or objects changing. Finally, we hope that in the future operating systems will provide much improved facilities for attaching attributes to files, such as meaningful names, keywords and other information, in much the same way as applications such as MS-Word allow these attributes to be stored, and subsequently queried. The new generation of object oriented systems such as CAIRO, OLE 2.0, NextStep, Open Doc and CORBA show promise, and future research will be directed to discovering to what extent these systems can provide the facilities we require.

ACKNOWLEDGEMENTS

The authors would like to thank Richard Crowder from the Department of Electrical and Mechanical Engineering at the University of Southampton for his help in producing the data for the case study, John B. Smith of the University of North Carolina at Chapel Hill and Ken Anderson of the University of California, Irvine, for sharing their experiences in this field with us, and to the whole Microcosm team at Southampton, particularly Ian Heath, upon whose system this research was based.

REFERENCES

[1] Anderson, K.M., Whitehead, E.J. Jr. & Taylor, R.N. *Chimera: Hypertext for Heterogeneous Software Environments.* In: The Proceedings of ECHT '94. Edinburgh. ACM Press, 1994

[2] Asfalg, R., Hammwöhner, R & Rittberger, M. The Hypertext Internet Connection: E-Mail, Online Search, Gopher. *In:* Raitt, D & Jeapes, B. *Eds: Online Information '93. 17th International Online Information Meeting.* pp 453-464. Learned Information Ltd. 1993.

[3] 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. 1992

[4] Engelbart, D.C. Toward Integrated, Evolutionary Office Information Systems. In: *The Proceedings of the 26th Joint Engineering Management Conference.* pp 63-68. IEEE, 1978.

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

[6] Greif, I. *Hypertext and Group-enabling: Lessons from the Desktop.* Key-note address given at Hypertext '93. Seattle, Washington. November 1993.

[7] Halasz, F. & Schwartz, M. The Dexter Hypertext Reference Model. *Proceedings of the Hypertext Standardization Workshop.* pp95-133, Gaithersburg. US Government Printing Office. Jan 1993.

[8] Hall, W. *Ending the Tyranny of the Button.* IEEE Multimedia. Vol(1)1. 1994

[9] Hammwöhner, R & Rittberger, M. *KHS - Ein offenes Hypertext-System.* Technical Report 28-93 (WITH-3/93), Department of Information Science, University of Constance. 1993.

[10] Jeffay, K., Lin, J.K., Menges, J. Smith, F.D. & Smith, J.B. Architecture of the Artifact-Based Collaboration System Matrix. In *Proceedings of CSCW '92.* ACM 1992.

[11] Leggett, J. & Schnase, J. *Dexter with Open Eyes.* Communications of the ACM 37(2) pp 77-86. Feb. 1994

[12] Li, Z., Davis, H.C. & Hall, W. *Hypermedia Links and Information Retrieval.* In: *The Proceedings of the 14th British Computer Society Research Colloquium on Information Retrieval,* Lancaster University, 1992

[13] Malcolm, K.C., Poltrock, S.E. & Shuler, D. Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise. *Proceedings of Hypertext '91.* pp 13-24. ACM. 1991

[14] Meyrowitz, N. The Missing Link: Why We're All Doing Hypertext Wrong. *In*: Barrett, E. *ed. The Society of Text: Hypertext, Hypermedia and the Social Construction of Information.* pp 107-114, MIT Press, 1989.

[15] Pearl, A. Sun's Link Service: A Protocol for Open Linking. *Proceedings of Hypertext '89.* Pittsburgh, Pennsylvania, November 1989. pp 137-146. ACM.1989

[16] Rizk, A. & Sauter, L. Multicard: An Open Hypermedia System. *Proceedings of the ACM Conference on Hypertext, ECHT'92.* Milan, Italy, December 1992, pp 181-190. ACM. 1992

[17] Salton, G., Yang, C.S. & Wong, A. *A Vector Space Model for Automatic Indexing.* Comm. ACM 18(11), pp 613-620, Nov. 1975

[18] Shakelford, D.E., Smith, J.B. & Smith, F.D. The Architecture and Implementation of a Distributed Hypermedia Storage System. *Proceedings of Hypertext '93,* Seattle, Washington, November 1993. ACM. 1993

[19] Vanzyl, A.J. *HyperTED Technical Description.* http://adrain.med.monash.edu.au/HyperTEDTechnical.html.