University of Southampton

An Open Hypermedia Link Service Architecture Supporting Multiple Context Models

by
Gareth Vaughan Hughes

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

August 2003

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS

SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

An Open Hypermedia Link Service Architecture Supporting Multiple Context Models

by Gareth Vaughan Hughes

Open hypermedia concerns itself with finding ways to provide links between information using systems that allow the direct manipulation of links. These systems are generally built in component fashion, a major component will be the Link Service. This thesis extends previous work on building link services for the World Wide Web in order to improve the quality of linking that can be provided. A link service delivers links into Web pages dynamically, this allows the system the opportunity to adapt the links to suit the user and content. The more general form of this adaption is to deliver links in some context. The core of this work is to examine how to generalise the delivery of links in context so that new context models and techniques can be introduced without writing a new link service each time.

This thesis examines the problem of defining models of context and how different models are often incompatible. It introduces a new architecture for exploring methods of providing links in context. The architecture consists of a core infrastructure along with a method for dynamically changing the components of the system concerned with the link model and the computation process for generating the final link to deliver.

The effectiveness of the architecture is explored through the undertaking of a number of projects using real-world data. The limits of the design are explored and the analysis focuses on how successful it is possible to be given the goals set.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Wendy Hall for being my supervisor over many years and for having the trust in my abilities to offer me a job with the group in the first place. My thanks must go to all the people in the group, past and present, who have helped me learn this new discipline. I did a degree in physics and did not learn a great deal about computing during it. I then joined the Multimedia Research Group (now Intelligence Agents Multimedia) and feel that the first few years here were the equivalent of doing a vocational degree in computer science. It was a degree done the hard way, on the job, and with a lot of people's help. With that completed I had the skills to produce the work in this thesis.

I wish to thank Steve Rake for reading the first full draft with such care and attention to detail. His suggested changes were to the benefit of anyone who attempts to read this thesis.

I wish to thank Lynda Hardman and David De Roure for being my examiners.

This work has been finally completed because I was given some flexibility by Wendy Hall and Richard Crowder. I am eternally grateful for the freedom and promise to pay Richard back in code and papers.

Special thanks to all on #iam_here, especially Bay 6.

This is for my wife, Jo, the most important person in the world.

1 Introduction

1.1 Motivation

For many people in developed countries the World Wide Web is now ubiquitous. It has become part of our daily lives in a way which could not have been imagined when I started my PhD 6 years ago. To the vast majority of users who have no idea that there have ever been other hypertext systems there is no significant problem with the Web. It is just the Web, a part of the infrastructure of the modern world. It has many wonderful properties that have made it so successful but there are still problems. For instance, those that have tried to build and manage Web pages know that it is difficult but there is so much money involved in the Web there are now plenty of commercial solutions, in fact a whole industry has sprung up where none existed.

The simplicity of the design of the Web is also a major design flaw. Links between pages are placed directly into the pages as part of the documents themselves. This makes it very easy to create simple Web pages and the ease of running a server makes it straightforward to make them available to the whole world. Anyone on the Internet can write a Web page and be part of a global community. This simple link model makes it a significant problem to manage just 500 pages.

My first experience of writing a significant number of Web pages was in December 1996 when I produced a Compact Disc based 'Web site'. The CD documented all of the projects being funded by an initiative of the UK science research funding council, HEFCE. The task was to download three or four pages of each project Web site, plus reports, and integrate them into a coherent whole. The CD included a categorisation of each project and a number of ways of navigating the contents. To do this, sets of pages were downloaded from the project Web sites, all of the internal links between pages were corrected to still work and my own navigation links added. Navigation links and graphics were also added to each and every downloaded page as part of the overall site design. All of this was done with a simple text editor, producing a complex structure many directories deep. There were over 500 files and even more links. Any changes made to the categorisation and hence the navigation of the pages meant editing every single page involved by hand to amend the links.

The task was so painful and repetitive to do that since then I have never produced normal HTML Web pages other than the occasional page.

The difficulties of working with raw HTML and the management of complex Web sites have led to it now being rare for people to edit HTML directly. There is now a mature Web publishing and content management industry. It is impressive to remember that it has largely grown from nothing during the lifetime of this PhD. One area that has not received the same attention is the generation of links between and within pages. For commercial Web site producers links are most commonly used as navigational tools, linking index pages or catalogue listings to data or content pages. It is rarer to see links within content of pages (Miles-Board & Carr, 2002).

Navigation links are the infrastructure of a Web site. Often they are mechanically created by software simply to be functional. Links within content can provide significant added value to documents if they are produced well. An author can cross reference readers to other interesting pages or pages that give greater explanations of technical terms. The links can refer to previous stories on a news Web site or link to other Web sites with a similar or differing opinion. Content based links bring the Web to life but require work to produce, careful thought in placing and maintenance to ensure that they continue to work correctly.

(Lowe et al., 1997) discusses this in detail and proposes that link authoring should be approached with the same discipline as software engineering. They claim that link authoring is in the same state as the software industry thirty years ago. Development is largely *ad hoc* with little proper thought given to process modelling or management of the production.

There are systems to help with this process. Tools have been produced to add links into content automatically. A simple but powerful application is one which will add a link on any occurrence of a word to a definition. For instance a Web site that includes many technical terms can produce a glossary section and link occurrences of the technical words to their definitions. These systems also need to be used judiciously if they are not have adverse effects. A simple mechanical search and replace technique to add links does not take into account the situation in which the document is being read, the users interests and will not be computed on demand for a particular set of circumstances. The links are not in context.

1.1.1 Open Hypermedia

The links in Web pages form an example of a hypertext, a field of study now over 50 years old. The foundation of which is the vision of Vannevar Bush in his post-war description of machines to aid us think and work 'As We May Think' (Bush, 1945). Many of the early hypermedia systems were built as single dedicated pieces of software encompassing all of the functions and data they required. As computers have improved the new possibilities have led to ever more sophisticated systems.

The hypermedia community has long advocated systems whose major property is their openness. There are many definitions of open hypermedia, one is that an *open hypermedia system is one which is open to the import of new objects* (Davis, 1995). This implies various requirements such as being able to support an unlimited size of system, to allow for new data types to be added, to allow multiple users have their own view on the system, to run on multiple, distributed platforms and to allow for differing views on the data model of a hypermedia application. Other definitions can be found in (Lowe et al., 1997) and (Østerbye & Wiil, 1996).

The Web is an open system because it is distributed and scales well, but it is a closed system due to the implementation of links (Lowe et al., 1997). A core component of open hypermedia systems is that links are stored, managed and served by a separate system called a link service. Links are not embedded or written directly into documents, as they are in Web pages, but stored in a database of some kind. This database, called a linkbase, is the source of links used by other components of an open hypermedia system. Chapter 2 contains a review of the areas of open hypermedia and link services.

This fundamentally different approach to implementing a hypermedia system opens up a much greater set of possibilities. A number of very powerful systems that had nothing to do with the Web were produced up to the mid 1990's using this approach. Now their number is dwindling (Bernstein, 1999) as research groups use the Web as the infrastructure for their systems and experiments.

When links are managed by a link service the possible uses grow and powerful effects can be produced. A link becomes a first class object to be managed, deployed and reused as required allowing for ease of management and efficiency. A link does not have to start at one place in a certain document and end at one destination. Links can connect multiple resources together which other components of the system can interpret as they wish. A link service can also be used to add links to content on demand in a multitude of ways. One such technique is to place a link on any occurrence of a word or phrase in any document in the system, and link it to a single destination. The classic usage is to link any occurrence of a word in any document to a page with a definition of it. These links are called generic links or glossary links (Carr et al., 1998).

The ability to add a link to any occurrence of a word in any document and link it to one destination is a powerful mechanism. Like any powerful tool if it is not implemented judiciously it can cause as many problems as it solves. The link might not be to the correct definition (for instance the word 'spring' has a considerable number of meanings) or the system may be confused by acronyms ('and' could be a word or a person's initials). A common problem that has arisen is that too many links are placed into the document giving too many choices and adversely affecting document presentation. To solve these problems the link service needs to understand more about the document, the user and situation in which the document is being used. In short the links need to be placed in context. This thesis presents a design for a system for doing just that.

1.1.2 Context

There are considerable challenges in determining and using contextual information at even the simplest level. The first problem to be tackled is to understand exactly what context is. The word stems from the latin for weave or entwine. This leads to one definition "The weaving together of words and sentences"[1]. Another definition is "The circumstances in which an event occurs"[2]. These cover the two major usages of the word. The first and older definition is on the subject of writing and defines context as the meaning given to words based on those around them. The second definition is slightly more general purpose and defines context as all of the factors that are needed to describe a situation or event.

We implicitly understand what context means in any given situation but building it formally into computer software is much harder. In computer science literature there are as many definitions of context as there are systems claiming to use it. Trying to reach a definitive statement of what context is and how to go about using it has been a difficult task.

This thesis describes work done in developing an architecture for an open hypermedia link service to add links to information in context. The design is such that new definitions of context and links with contextual information can be designed and added to the system along with any special methods required to process these links. The system has been used in the implementation of a number of

[1]Oxford English Dictionary.
[2]The American Heritage Dictionary of the English Language, Fourth Edition.

hypermedia applications. These demonstrate that a variety of simple contextual situations can be modelled and that the architecture is flexible enough to accommodate them. The variety of experiments allow an evaluation of the design which considers the limits of what is possible and how to proceed from this point.

1.2 Contribution

The work presented in this thesis contributes to the area of open hypermedia. In particular the major contribution is the development of an architecture to allow the changing of the link service used by a hypermedia system according to a form of context. The other contribution is to survey and examine the meaning of context as used by different fields of computer science. The survey finds that there is no correct solution to properly defining context and hence the system presented here allows for interchangeable models of context to allow for new methods of providing links.

1.3 Thesis Structure

Chapter 2 presents a review of previous work on open and adaptive hypermedia systems along with the development of the hypermedia link service as an independent entity.

Chapter 3 describes the development of a document management system and the integration of an open hypermedia service. The integration was taken even further through a number of experiments. The aim was to improve the quality of the links provided by adding links in context. This leads to the conclusion that a new design of link service is needed that specifically copes with context.

Chapter 4 reviews the incredibly difficult question 'What is context?'. It looks at how context has been a core problem in the field of Artificial Intelligence for decades. The review looks at how context is represented, how changes in context can be determined and how context can pragmatically be used by computer programmes. It examines how any model of context will always be flawed or broken by some unexpected factor which leads to the conclusion that any system incorporating context needs to be able to use multiple models of context.

Chapter 5 describes the design for an experimental architecture for resolving links in context. This link service allows new methods of resolving links to be loaded and new context models to be added on demand.

Chapter 6 describes the major implementations of the system in work undertaken with an external sponsor. In the first phase of the work a system is built to add links to documents which depend on which document the reader was viewing in a 'document space'. A context model was developed linked to document location. It then describes how the architecture was used to create a method for generating new documents based on the existing corpus of information. The system used a new model of contextual link processing to extract paragraphs for the document corpus and combine them into new documents.

Chapter 7 explores the limits of the system design. This is done through the building of a number of other context models and ways of resolving links. For example a system that understands the age of a document in an archive and can generate links appropriately, perhaps for a news site. Another is a system for adding generic links into works of fiction to give a character summary, but the service knows where the reader is in the book and only reveals the plot information they would have read at that point in the story.

Chapter 8 concludes this work and draws the various strands together. It examines whether the proposed architecture is a success and discusses methods to improve the overall model used. Future work is proposed to tackle weak points in the design.

1.4 Declaration

This work has been done within a collaborative environment. The work described in this document is the sole work of the author unless specifically credited to others.

2 Hypermedia Linking and Adapting

2.1 Open Hypermedia Systems

2.1.1 Hypertext Origins

A prime point to start a survey of hypertext systems is with Conklin's 1987 survey of early hypertext systems (Conklin, 1987). In it he maintains that machine-supported links are the key distinguishing feature of a hypertext system. Other features such as a window-based viewing system or text processing capabilities are shared with many other families of applications that are not necessarily hypertext systems. The paper summarises many of the early, classic, hypertext systems including Bush's visionary ideas as incorporated in the Memex (Bush, 1945), NLS/Augment (Englebart, 1963) and Xanadu (Nelson, 1987) before giving Conklin's view on the advantages and disadvantages of hypertext. The term hypertext and the term hypermedia are attributed to Ted Nelson and refer to nonlinear text for navigating an environment (Nelson, 1987).

Conklin starts by giving a definition of hypertext. Hypertext allows a user to easily trace references, given within documents, by the form of a link which can easily be followed forwards and, occasionally, backwards. A user can add to the network of ideas by being able to easily create new links or add annotations without altering the original documents. A variety of structures can be imposed onto a collection of documents that may not already have a structure. The majority of systems allow a non-hierarchical structure to be built whilst some early systems had specific support for creating hierarchical structures. Systems can customize documents by linking particular sections of a document together allowing for efficient reuse of material. This increases the modularity of the information in the system leading to a tendency to split larger documents into smaller modules to make them more useful.

He also states that a powerful feature of hypertext systems is that as the links were embedded in the text of the documents the text could be moved around and the links would still work. This is an argument that has to be seen in the context of the age of the paper. The majority of systems developed since the mid 1980's, with one notable exception, the World Wide Web, have advocated the separation of links from the content of documents. Conklin also cites the ability to explore several paths of enquiry through material and to be able to trace back through the path using history mechanisms.

Conklin describes two main problems with hypertext systems. Disorientation and cognitive overload. The ability to organise information in a much more complex manner also makes it much harder for a user to successfully navigate the information space. This problem increases with the number of nodes. Cognitive overload arises from the extra work the user needs to do to keep track of their location in the document space. This thesis concerns itself with ways to tackle these problems through better targeting of links and content to the reader and the context within which the document is being read.

2.1.2 The Evolution of Link Services

Up until the end of the 1980's hypermedia systems were largely monolithic applications. The functionality of both data or document management and link management were provided by the system as a whole. There was no attempt to separate the link functionality to cope with changes in link design or other such updates. There were calls to change this and provide hypertext functionality to other applications (Meyrowitz, 1989).

2.1.2.1 Intermedia

An early hypertext system to have a separate link server was Intermedia (Yankelovich et al., 1988). This system used a link server which communicated to the rest of the hypermedia system using socket connections. The rest of the system was custom built and the authors did not try to integrate the link service with outside applications.

The ability to link between different types of nodes raised a number of issues that many open hypertext systems have since faced. The link service had no direct way of knowing if a node had been deleted without performing an exhaustive search. There was a similar problem caused by changes to a document that already contained links. In order to know where to put a link within a node the link service assumed that the linked object was a single full line of text. This text was stored as part of the link data and when the document was loaded the service searched the document to find the first occurrence of that line. A link was marked in the document by a simple icon which the application developer needed to place. This simplistic approach only worked if the system was used for text documents. Because the system was designed to work with any application the authors did not design specific link types or attributes, including directionality. The authors felt it better to allow link types to be created dynamically rather than create a fixed set which may limit the usefulness of the design.

2.1.2.2 Sun's Link Service

The first practical implementation of a link service and the first use of the name was in Sun's Link Service (Pearl, 1991). This was not a hypertext system in its own right but a protocol and application developers toolkit to add hypertext functionality into existing applications on a Sun workstation. The implementation consisted of a link database and a library which a developer added into their application. The library implemented a series of windows and dialogues adding hypertext functionality to an existing application without having a large impact on the existing program's interface.

2.1.2.3 PROXHY

The PROXHY[1] system (Kacmar & Leggett, 1991) was a prototype implementation of an open hypermedia architecture design based on a process and object oriented model. In the model anchors, links and applications were separate actors communicating through a message passing protocol. It was a significant step forward in terms of open hypermedia design. The architecture consisted of four major components: hypertext, communications protocol, application and back-end. The hypertext layer comprised the anchor and link processes which, like all parts of the system, were written as classes and implemented as processes. The communications protocol layer supported networked communications allowing the system to be distributed. The back-end layer provided data storage services to the applications layer.

An application needed to implement support for the communications protocol and also to support a global unique naming convention for objects. As the whole system was based on an object oriented design it was a relatively easy task for a developer to utilise the existing classes for these functions. The ambitious design gave considerable flexibility but at a cost of large amounts of message traffic resulting in poor performance.

2.1.2.4 Multicard

The Multicard hypermedia system (Rizk & Sauter, 1992) was a serious attempt to write a hypermedia system that could be used on a large scale in an industrial environment. It featured an extensible protocol (M2000) for sending messages between different processes in the system. The provided hypermedia toolkit consisted of a scripting language, basic tools for creating and manipulating distributed hypermedia structures, an interactive authoring and navigation tool as well as the protocol

[1]Process-Oriented Extensible Hypertext Architecture

for applications to communicate with a back end providing the hypermedia services. In order to participate in the system an application only needed to support a basic subset of the protocol and a developer could choose what level of integration to implement.

The system featured an interesting approach to link usage. Links were event/message communications channels between the two end points of the link. The default message being one to activate the destination object. Link endpoints could be anchors, nodes or groups of either. It was possible to attach scripts to any of these objects. In effect the endpoints of a link became handles or ports to use in the scripts. This powerful ability meant that the system could cope well with evolving system or link design and could adapt to new editors being introduced or changed.

2.1.3 The World Wide Web

It is necessary at this point in the story to introduce the World Wide Web as it is the system against which all others must be compared. The utter success of the system and the revolution it has caused not only to hypertext research but to the world in general means it needs to be given special treatment in this document. Whilst I shall not explain the basics of the system in any great detail, as this should not be necessary to anyone, I will examine the design of links in the system as their simplicity is the root of the work described in this thesis.

The World Wide Web is a distributed hypermedia system. In it documents are written in a format called HTML[1], itself derived from SGML[2]. This allows authors to embed links to other pages by means of a syntax for referencing the page and server on which it is located. The source of the link is normally underlined in blue by the browser indicating that it can be activated by a click. The click will cause the browser to fetch and display the page. The behaviour can be augmented in numerous ways as competition has led to companies adding more and more features to the browsers. Scripting languages now give considerable control over link activation behaviour, this is usually used to provide features such as drop down menus on links for navigation uses. The main way to expand the capabilities of a browser is to download and install application plug-ins. These normally give the user the ability to view material created with proprietary systems such as Apple's Quicktime or Macromedia Shockwave. The price is that browsers have become very large and monolithic in nature. Only a few organisations have the ability to produce a modern browser capable of supporting all of the formats in use today. This has led to problems of monopoly in the browser market.

[1]Hypertext Markup Language
[2]Standard Generalised Markup Language

Whilst not an open hypermedia design the system is able to scale enormously and is highly distributed. The simplicity of the basics of producing a Web site and making it available to everyone on the Internet meant that the system was bound for success. When browsers became high quality and freely available the revolution was guaranteed. The Web has had an effect on the Hypertext community too. It is now rare to see hypermedia systems research that is not directed at generating Web pages or augmenting the user experience of the Web, usually by adding browser functionality.

2.1.4 Hyperwave

One of the major hypermedia projects of the 1990's was Hyper-G (Andrews et al., 1995). This ambitious system was designed to solve many of the problems of previous systems as well as break new ground in its support for multiple languages and its scalability. Support for the World Wide Web was added and is now a commercial product, Hyperwave (Maurer, 1995). The system is a scalable, distributed, open hypermedia system comprising servers and clients. The underlying system is an object-oriented database in which documents and links are grouped together in hierarchical collections that each user maintains in their own private space.

The system has all the features of the UNIX file permission system and allows for complex information structuring and link maintenance. The original versions of the system made use of a number of specially written clients and a proprietary rich text format. The commercial system is now designed to store a wide variety of common document formats. Full index searching is automatic for all documents and powerful searching is possible within documents, collections and across multiple servers. The strong language support includes translators to make links available in any language. A special collection type called a cluster allows for clustering of multimedia composites or multilingual aggregates. This allows the display of different language versions of the same document where required.

The core of Hyperwave is an object oriented database. Links, in the form of source and destination anchors are part of the object hierarchy supported by the server. This allows the system to serve and maintain links like any other object in the system as well as provide all of the tools that a proper database design brings. Links are bidirectional, the source of a link can be found from the destination. This allows the system to have link consistency, it is possible to deal with the deletion of a document from the system by not showing links to that document and guarantee that there will not be 'dangling links'. A protocol allows multiple servers to maintain referential integrity for all objects including links.

2.1.5 Microcosm

Microcosm was begun before the Web existed as a project to integrate a variety of sources of information into an open hypermedia system and provide a researchers workbench system within which to explore new ideas. (Fountain et al., 1990) explains the main design aims for the open hypermedia system.

No distinction between author and user. Anyone could make links within the system and removing the idea that link making was an ability limited to administrators or teachers.

Modular architecture. Software components could be replaced allowing individual projects or researchers to develop new components to augment the behaviour of the whole system.

Loosely coupled system. The design called for a multiprocess system with no interdependencies between the subsystems. When Microcosm 1.0 was developed on Microsoft Windows 2 this was an ambitious goal.

Links separate to documents. On top of the other benefits already discussed designing a system where the links were separate entities would allow the development of tools to analyse and manipulate links. Links were stored in link databases, known as linkbases. Multiple linkbases could be used simultaneously.

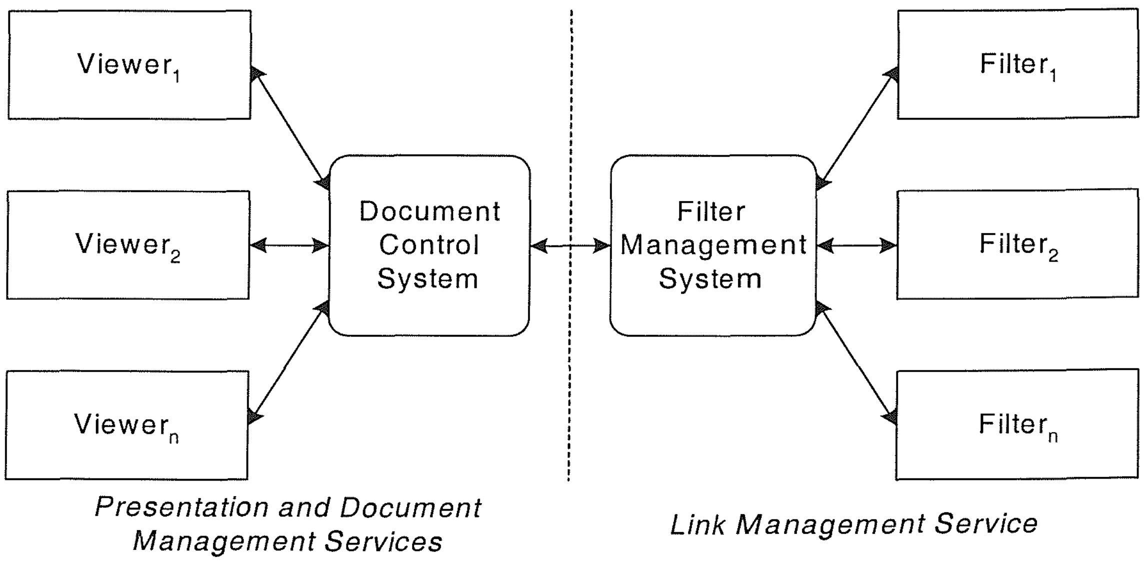


Figure 2.1: The Architecture of Microcosm.

The core of the Microcosm architecture comprised two communicating processes. One, the Document Control System (DCS) managed the front end display components. The other, the Filter Management System (FMS) managed communication between the hypermedia link processing components.

The DCS managed the launching and communication with document viewers. A document viewed in Microcosm was viewed in a dedicated viewing application which would communicate with the DCS. The message system in Microcosm was extensible and allowed components to alter any part of a message using a standard method. The FMS managed communication between the components of the system that provided hypertext services. The subsystem's primary function was to respond to and augment messages flowing through the system, hence the name 'filters'. Messages flowed in a single line along the line of filters in the 'chain' before reaching the Message Dispatcher (Hill et al., 1993).

Figure 2.2 shows the Filter Manager component of Microcosm. The right hand pane shows all of the available filters and the left hand side shows the actual filters in use within the running system.

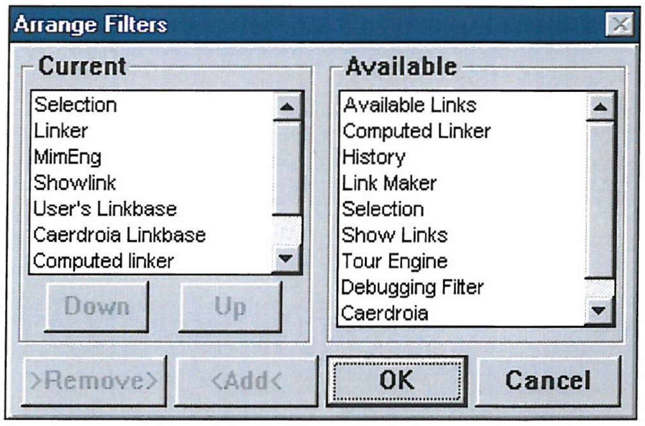
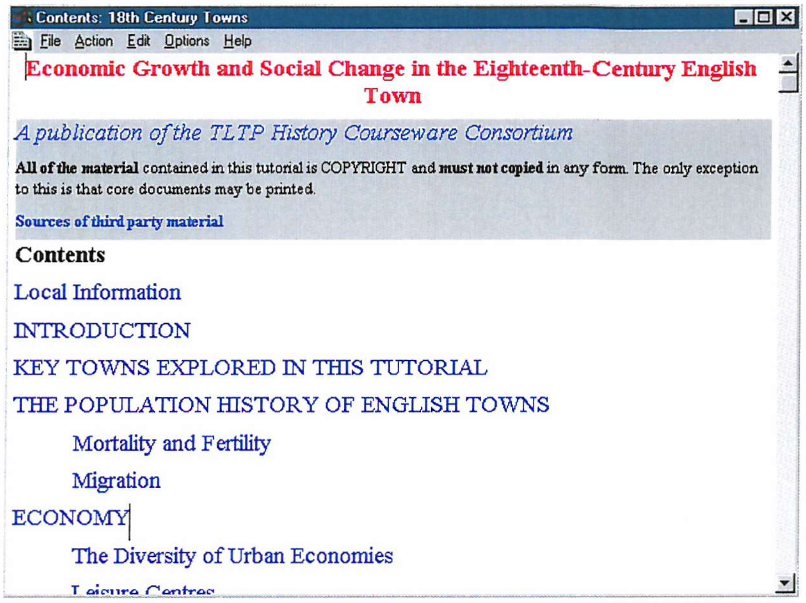


Figure 2.2: The Filter Manager Component of Microcosm.

A user interacted with documents in the system through dedicated pieces of software called Viewers. The level of integration of the viewer with Microcosm led to the notion of fully-aware and partially aware viewers. A fully aware viewer was a piece of software specifically written for Microcosm and could support the full range of hypermedia functionality. Figure 2.3 shows a typical document viewer. The Action menu gives access to the hypertext functionality in the system.



Economic Growth and Social Change in the Eighteenth-Century English Town

A publication of the TLTP History Courseware Consortium

All of the material contained in this tutorial is COPYRIGHT and **must not copied** in any form. The only exception to this is that core documents may be printed.

Sources of third party material

Contents

Local Information

INTRODUCTION

KEY TOWNS EXPLORED IN THIS TUTORIAL

THE POPULATION HISTORY OF ENGLISH TOWNS

Mortality and Fertility

Migration

ECONOMY

The Diversity of Urban Economies

Leisure Centres

Figure 2.3: A Microcosm Document Viewing Application.

The linkbase filter provided the core link database functions for creating new links, following link actions and returning all of the links for a document.

A number of types of links were used in Microcosm.

Specific Links. A link from an object in a specific location in a specific source document. The location could be defined using an offset in text documents or as a region in images. The destination of the link could also be a specific object in the destination document. A button link was a special case of this form of link. In this case the source selection would be coloured blue. All other links were not visible in the source document.

Local Links. A link on a certain word, word-pair or object at any location in a specific source document. The link can be followed from any occurrence of the selection in the source document.

Generic Links. A link on a certain word, word-pair or object occurring in any location in any document. A common usage for such powerful link design was as a glossary link, to link a word to a document giving its definition.

When a user opened a document in the system a specially written document viewer would request links for the document from the system. The linkbases would return found links that would reach the end of the filter chain before being returned to the viewer. It was a relatively simple matter to write a filter to alter the messages and hence the behaviour of the system. For instance a number of interface components were written for the system that knew the documents that were open and the user's browsing history (Bernard et al., 1994, Hall et al., 1996). One use of such a tool would be to stop a new copy of an already open document being spawned and to bring the existing window to the user's attention. This involved the crucial ability to act upon and alter the results of the link services depending on the user's circumstances. Figure 2.4 shows a toolbar written to help users manage the multiple window interface of the system and provide shortcuts to key documents in the system.

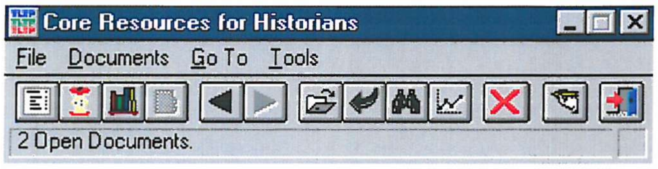


Figure 2.4: A Toolbar for Helping Users Manage the Microcosm Interface.

As part of the goal of building an open hypermedia system, work was done in investigating the levels of integration and hypertext awareness that could be brought to other applications on a user's desktop. The goal being to increase the move away from the early monolithic hypertext systems and towards a point where the hypermedia services extended to all applications. Achieving this led to the development of the notion of light hypermedia services (Davis et al., 1994, Knight, 1996).

A number of levels of integration were identified, from specifically written viewers to the worst case where a hypermedia system could only launch a third party application with an appropriate file and expect no further interaction.

Tailor made viewers. A specifically written part of the hypermedia application and fully integrated into the system. Microcosm featured a number which over time grew in power and complexity but cost a great deal in terms of hours of programming and support.

Source code adaptation. If the source code of a third party application was available then the hypermedia functionality could be added.

Application interface adaption. Many applications featured scripting interfaces which allowed features to be added such as extra menu items.

Shim or proxy programs. A program to sit between the application and the hypermedia system to augment or enhance the communication between the two. Examples included the Microcosm Universal viewer, described below and the Web proxy implementations of a link service such as the Distributed Link Service, described in Section 2.3.2.

Launch only viewers. In the worst case the hypermedia system would only launch a third party application with a data file. The application had no hypermedia functionality and no further interaction with the rest of the service would be expected.

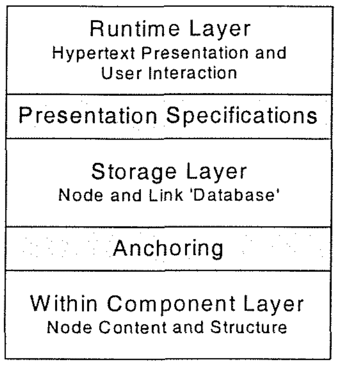
The Universal viewer was an attempt to integrate any application on the users desktop with Microcosm. The program behaved as a parasite to the main application by adding new buttons to the title bar of an application. These gave access to the available hypermedia functionality. For instance if a user clicked on the Follow Link action on the Universal Viewer (UV) the UV would attempt to ascertain the current selection of the host application, for example a text selection, and send that to Microcosm in the standard Follow Link message.

2.1.6 Open Hypermedia Reference Models

A constant theme in the history of open hypermedia systems has been the attempts to integrate the systems together and to find reference models as a basis for interoperability. The first of these, a highly influential piece of work, was the creation of the Dexter model.

The Dexter model was the result of a series of workshops in 1989 between researchers from many of the main US hypertext groups (Halasz & Schwartz, 1994). It was realised that there was little interoperability between systems of the time and the model was a way to create a shared understanding and reference point of the components required.

The Dexter model divides hypermedia systems into three main layers.



Runtime Layer
Hypertext Presentation and User Interaction
Presentation Specifications
Storage Layer
Node and Link 'Database'
Anchoring
Within Component Layer
Node Content and Structure

Figure 2.5: Dexter Hypertext Model.

The runtime layer is concerned with the presentation and manipulation of the system. It deals with the interactive aspects of the system. The storage layer is the focus of the design. It is a database of components, often referred to as nodes in many systems, between which there are links. Links are entities that store relationships between components. They feature end point anchors, direction and a presentation specification. The within-component layer stores the data about the structure and content of nodes. It is largely unspecified as to how this will be implemented.

Between these three main layers there are two others. The anchoring layer specifies how to address a specific location or item within an individual component. The presentation specifications layer encodes how a component is to be presented to a user.

The design has been very influential and systems are still being produced to conform to the model. Dexter is not without its shortcomings and there have been numerous contributions that improve on many aspects of the design. Two groups in particular have continued to work on systems and to extend the model. DHM[1] is considered to be the system that is closest to the principles of the Dexter model. It has extended the model into the areas of cooperative working (Grønbæk et al., 1994) and more recently as a system for the Web (Grønbæk et al., 1997). The other major system closely following the principles of Dexter is AHAM[2], described in Section 2.2.5.

By the end of the 1990's the major hypertext groups were realising that there were a number of highly advanced open hypermedia systems but each was unable to use any of the resources or components of the other. There began a long series of interoperability workshops, the Open Hypermedia Systems

[1]Devise Hypermedia Model
[2]Adaptive Hypermedia Applications Model

(OHS) workshops (Reich et al., 2000). The starting point to this was a call for a protocol for linking, OHP[1] (Davis et al., 1996), which proposed a method of protocol conversion between the various link services and client applications. This protocol was flawed but formed the basis for the OHS Working Group's meetings.

The group used various scenarios to produce a list of requirements for their overall framework.

1. Standardized representation format.
2. Means for identifying structure.
3. Integration of arbitrary data sources.
4. Update structural information.
5. Support for multiple structures.
6. Support for combining multiple structures.
7. Provide a consistent user interface across applications.
8. Support for references between distributed document partitions.
9. Support for access control.
10. Support for group awareness.
11. Support for joint editing sessions.
12. Support for consistency mechanisms between document partitions.

Following on from this they developed a unified data model drawing on the many similarities between many of the models of the existing systems. The core data model is presented below in Figure 2.6.

[1]Open Hypermedia Protocol

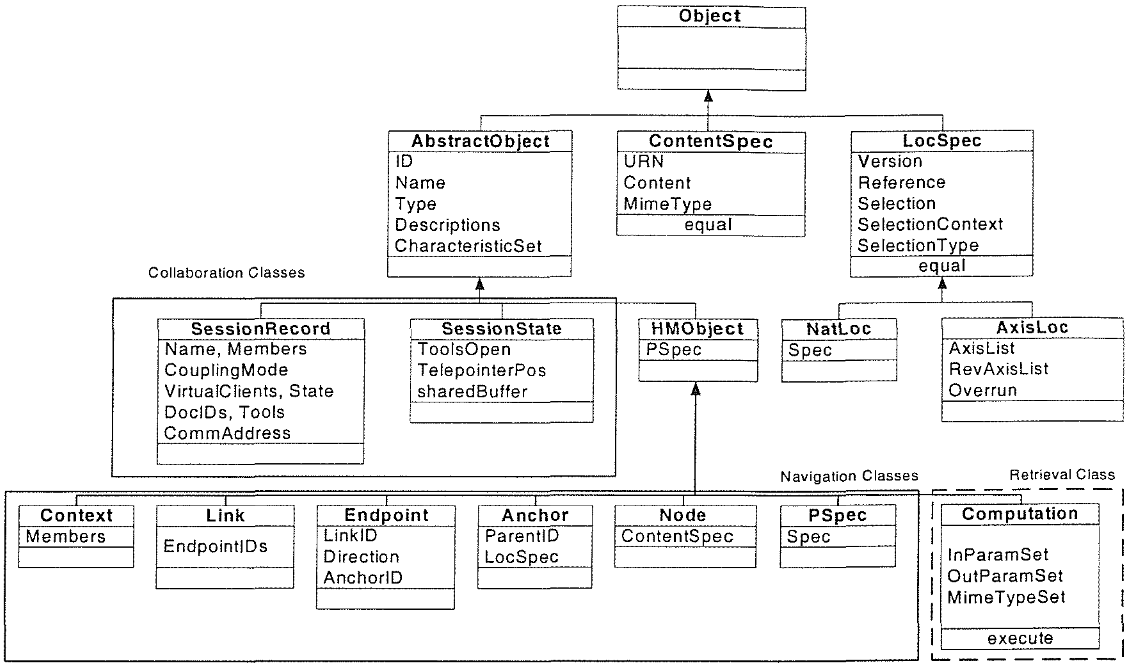


Figure 2.6: OHP Data Model.

The group produced a number of demonstrators of the principles including the Solent system (Reich et al., 1999) and the Construct system (Wiil & Nurnberg, 1999). From there the group has evolved and the workshops have continued. One result of the work was that the core data model was extended to become FOHM[1] (Millard et al., 2000).

2.2 Adaptive Hypermedia

In order to provide a better, more adaptive, link service the system must take account of the situation in which it finds itself. This is a field in which a lot of work has been done, the majority of it aimed towards providing educational systems designed to cope with the student's progress. This field is called adaptive hypermedia.

An adaptive hypermedia system provides functionality that is personalised for a user. The system builds a model of a user's goals, preferences and knowledge in order to improve the experience for an individual. There are many approaches to building such systems that concentrate on various

[1]Fundamental Open Hypermedia Model

aspects of the user model or the technologies to deliver the systems. The definitive classification of such systems and the fields of work is Peter Brusilovsky's 1996 survey of the state of the art at that time (Brusilovsky, 1996).

He begins the survey with a working definition:

> "by adaptive hypermedia systems we mean all hypertext or hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user"

The key characteristic of an adaptive hypermedia system is that such a system is based around the use of a user model as the root of adaption. The basic process of the system consists of three stages. The system collects data about a user, the data is processed to form a user model, the model is used by the system to effect adaption.

Brusilovsky reviews the methods and techniques for providing adaption. He produces the following questions to ask of an adaptive system design.

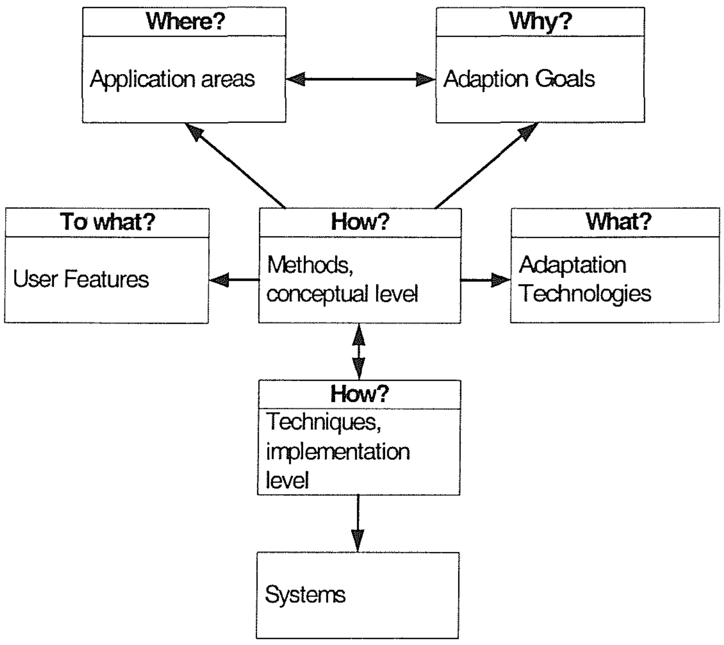


Figure 2.7: Major Questions to Ask of an Adaptive Hypermedia System (Brusilovsky, 1996).

These fall into 4 major sections. Within these sections Brusilovsky goes into some depth examining the various topics and reviewing a considerable number of systems of the time.

Where are adaptive systems helpful?

In what situations are adaptive hypermedia systems being used and for what purpose.

What features of the user can be used as the source of adaption?

What aspects of the user working with the system can be used when providing adaption.

What are the goals of adaption?

Where the techniques can be used, why they are useful and the methods which can be employed.

What can be adapted by a particular technique?

This can be generalised into techniques for either content or navigation adaption.

In each of these areas it is possible to focus and expand greatly. We shall examine each in a little detail but concentrate on the areas most relevant to designing a link service architecture: techniques of adaption.

2.2.1 Where are adaptive systems helpful?

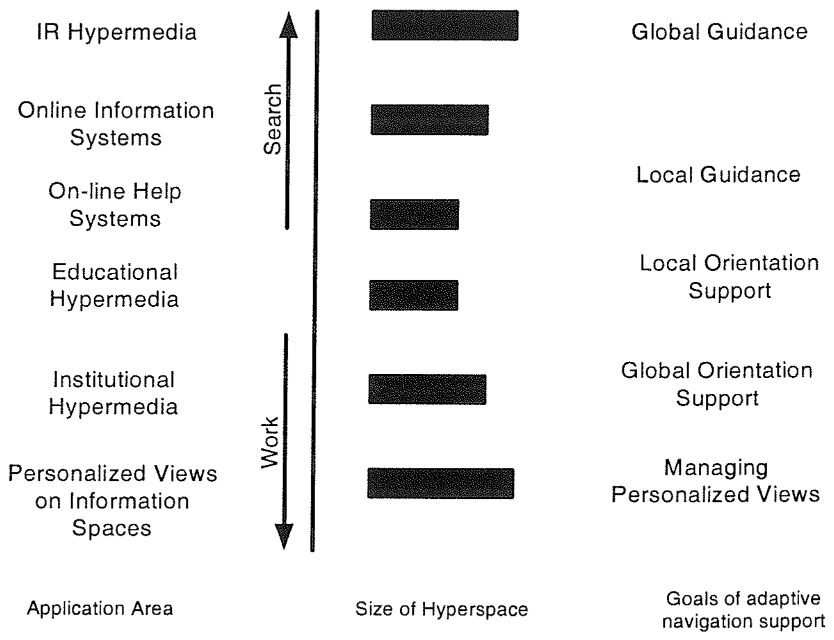

IR Hypermedia
Online Information Systems
On-line Help Systems
Educational Hypermedia
Institutional Hypermedia
Personalized Views on Information Spaces
Search
Work
Global Guidance
Local Guidance
Local Orientation Support
Global Orientation Support
Managing Personalized Views
Application Area
Size of Hyperspace
Goals of adaptive navigation support

Figure 2.8: The Range of Adaptive Hypermedia Applications (Brusilovsky, 1996).

Adaptive hypermedia systems are mainly used to provide support in the following ways.

- Providing navigational guidance, perhaps to novice users in an educational system or by reducing choice in a large information retrieval system.
- To personalise a system for a user, perhaps in a way that reflects their use of the system for their work when using an intranet system or their information seeking goals when using information retrieval systems.
- To adapt to the users level of experience in using the system in question. If the user is new to such systems then they can be guided and have their choices restricted thus keeping the interface simple. An experienced user needs to move quickly throughout the application to find what they need. In one case the system could be an educational hypermedia application whereas the latter is more likely to be in a work related situation where the person uses the system regularly as part of their job.

2.2.2 What Features of the User Can be Used as the Source of Adaption?

In order to provide adaption a system must have criteria upon which to base its decisions. These mainly derive from some form of model of the user. The most common implementation is to represent the users knowledge as an overlay model which is based on a structural model of the domain. These are often quite sophisticated systems consisting of a network of domain concepts. These can be linked together in a semantic network. Both the model and adaptive system need to be able to cope with the user model changing whilst in use. For each element of the model the system holds some type of value of how well the user knows the concept. These could be binary, from a fixed list of possible values, or a quantitative value such as a percentage.

A simpler method is to use a stereotype model. These involve assessing how closely a user matches a listed set of stereotypical users for each dimension of the model. These could encompass ranges such as "novice - expert" or "junior rank - high rank". Even simpler systems might utilise a true/false or probability value for whether a user belongs to a certain stereotype. Such systems are simpler but much easier to implement and maintain.

Often a combination of the two models is used. The stereotypical model is used to start the user with the system and can be filled in following a short interview. From here the system uses these values with a full overlay model.

A user model is different to user preferences, which can also be used by such systems. The system cannot deduce preferences itself but must obtain them directly from the user. The representation of preferences also differs in that model data is often stored symbolically but preference data is stored as numerical values. The obtaining of these values can be achieved by highly complex methods. Because the data is numerical it is possible to combine several user models together to form a group user model. Preferences can sometimes be the only data stored for users of some systems. Many information retrieval systems just use such data.

In a similar fashion to using a model of the user's knowledge of the subject, two other features can be used, their background and experience in the given domain. Background is relevant information about the user from outside the hypermedia system. This could be factors such as the user's profession or previous work in the subject area. Experience is a measure of how familiar the user is with the system in question rather than the subject.

Task or goal oriented features relate to the user's work when using the system. Goals can range from high level overarching educational goals to rapidly changing goals such as learning each new piece of material in an education system. The use of such models is usually to influence the navigational aspects of a system. Many are modelled in the same way as overlay models with some quite sophisticated techniques in use. These include a hierarchical tree of possible goals or using a probability value against a list of goals as to which goal is the most likely to match that of the user.

Before a profile or model can be used it must be captured and processed before being applied to the system. Capturing data for a model is a balance between automated methods and manual input from the user. In the case of most adaptive hypermedia systems fully automated methods are limited due to the sources of information available to system builders. The most obviously available sources are the time the user spends reading a particular node and the trail through the application. Neither are particularly reliable to use, a person may spend a long time on a particular page but may not be reading it.

Therefore additional sources of data are required in the form of direct user input. In this there is a balance between input methods that are quick and simple, such as a way of directly indicating how relevant a page was to the user, and longer but more accurate methods such as providing an interface to the user model itself and letting the user edit it. In between these extremes are methods such as allowing the user to move hypertext elements around the screen to suit themselves, for instance re-ordering navigation links or hiding/revealing pieces of text. To supplement this the user can be directly asked to provide specific information about themselves to feed the model. In all cases a design decision must be made to balance quality of data required against distracting the user to the point where the user resents supplying the information.

2.2.3 What are the Methods of Adaption?

In this section we look at some of the overall methods by which adaptive hypermedia techniques can solve some of the problems found in hypermedia systems. The next section goes into detail on some of the implementations used.

Content adaption is most usually associated with hiding information from users or steering them away from information that is considered irrelevant. This can work both ways, hiding advanced information from novices or basic information which is considered superfluous for advanced users. Other techniques include using pre-requisites and comparative explanations. Pre-requisite information is that which is considered the user must have seen before viewing the current piece of information. If they have not read those pieces of information then links are given to them. Comparative explanations work similarly to link pieces of information together to compare or contrast topics. Both techniques are most commonly used in the teaching of programming languages or other subjects where the knowledge must be learnt in a structured way. Beyond that the decision over what to show can be tied to the user model in some way. In some systems the fragments of information can be sorted to alter the priority of them for a user.

Navigation adaption methods aim to achieve two main types of goals. To provide guidance in navigating the system, either making sense of the overall system or to help decide where to go next. The other function is to help the user orient themselves in the system, again either in the global sense of the system or at a local level so that the user knows what other nodes are around the current one. Methods include hiding or sorting navigation links, dynamically generating a 'next' button to directly guide a user and using a user model or goals to influence the navigation layout. Navigation guidance between local nodes can be improved with techniques such as giving an indication of the nature of the destination, by ranking, colouring or rating the links.

2.2.4 What can be Adapted by a Particular Technique?

(Brusilovsky, 1996) categorises the implementable techniques of adaptive hypermedia into two main areas, adaptive presentation; the altering of the content of a node, and adaptive navigation; techniques for guiding the user around the system.

Adaptive presentation is most normally associated with the adaption of text content to suit the user, more rarely is it concerned with multimedia adaption. Navigation techniques include direct guidance, link sorting or hiding, link annotation and map adaption.

Map adaption is the altering of images that provide navigation assistance to the user, these are now most commonly found on the Web as image maps with clickable areas. It is still rare for these to be implemented in an adaptive fashion on most Web sites.

As previously discussed direct guidance aims to provide a best direct route through the material for a particular user, for instance by generating a 'next' button according to a user model. Sometimes such systems can fail if they make no provision for the possibility that the user does not want to follow the precscribed route.

A more flexible solution is to re-order links to suit a user in order to suggest a best route. This leaves more options open but gives clear indications where the user should go in the system. The other related technique is to hide links considered irrelevant to the user. These techniques can lead to the problem of breaking the user's mental map. If navigation links are continually being reordered or disappearing then the user cannot build up familiarity with the system.

Link annotation is a newer solution to these problems. Links are annotated with information about the destinations. These can take the form of textual annotations, such as popup text boxes, or in some other visual clue such as a colour to indicate a link priority e.g. a traffic lights colour scheme.

2.2.5 Key Adaptive Hypermedia Systems

This section describes some of the key adaptive hypermedia systems and projects of recent times. They illustrate many of the methods and techniques outlined above but angle more towards the particular issues that arise when producing such systems for the Web.

Adaptive hypermedia systems that were created before 1993 are generally referred to as First Generation systems. They were single user systems lacking distribution and open hypermedia functionality. First generation systems tended to be monolithic in nature. Second generation adaptive hypermedia systems are generally targeted at the Web and add features such as greater openness of design and greater distribution.

A major first generation system was Hypadaptor (Hohl et al., 1996), used to teach the programming language LISP. The goal was to take account of the highly personal way in which programmers learn and work. Hypadaptor was a system that could dynamically adapt content presentation and navigation to a sophisticated user model of the student. Learning material was also modelled and stored in a structured knowledge base. In order to initiate the user model the student filled in a questionnaire about their programming skills and knowledge of Lisp. This fairly course system was enough to start the process of adapting to the user's needs. The knowledge structure, object oriented in nature, was navigable using a topic browser which could be used within documents or as an aid to navigation around the structures. This highly structured knowledge base was the key to the system. With this close integration between system, knowledge base and user model the developers could provide many adaptive features to aid learning.

-

One of the major pieces of work in this field is the AHAM (Bra et al., 1999) system based on the AHA (de Bra & Calvi, 1998) architecture. It combines the classic principles of an adaptive system; user model, domain model, pedagogical model, with the Dexter model. In the model the Dexter storage layer is split into the domain, user and teaching model parts. This allows the requisite adaption based on the domain and user model using pedagogical rules. As with the majority of adaptive systems the major uses for the systems based on the model have been educational but the applications have also developed into general purpose tools for Web site generation (Wu & Bra, 2002, Bra et al., 2002).

A good example of a second generation adaptive hypermedia teaching system is Interbook, a system for creating adaptive electronic teaching books produced at Carnegie Mellon University (Brusilovsky et al., 1998, Brusilovsky & Pesin, 1995). The system uses two models. The first is a domain model of the concepts to learn. In this case the pages of an Interbook book can be indexed by the author. Concepts could either be pre-requisites or outcomes. Pre-requisite concepts are those the student is required to know before proceeding with the current page in the book. An outcome is a concept that the student will now understand following the reading of a page or some other task such as passing a test. The system implements sequences of concepts for a student to learn. Such models were developed much earlier in the field of Intelligent Tutoring Systems (Polson & Richardson, 1988). The second model records the progress of the student using the system.

The delivery of material is through the Web with the implementation based on a Lisp-based Web server Common Lisp Hypermedia Server CL-HTTP. The authoring of material is facilitated by writing in Microsoft Word and using a structured template to extract the content into a structured and specially marked-up version of HTML. The process allows users to add the concept-based annotations necessary to place the material into the concept model. The concept model provides the basis for the generation of navigation links between concepts but is also the foundation of a glossary. The glossary matches the network of domain concepts so there is a glossary entry for each concept in the model.

During evaluation experiments it was shown that people did not use many of the components, for instance hardly any users had ever read the online manual, itself presented through the system (Brusilovsky & Schwarz, 1997). Therefore it was decided to introduce the interface elements to the user during their reading of the manual. The starting interface was a simplified one and as the user read through the online manual the newer interface elements were revealed to them. They chose to implement a model in which each user interface component was considered a unit of learning in a teaching model. The user became a student learning to use the system.

Interbook is a successful and mature system in use today. It is distributed and large scale but is not an open hypermedia system. It uses its own form of adapted HTML pages and complex domain model to link concepts of learning together. Navigation linking is an implicit part of this structure as is glossary linking. When combined with the student model the system is flexible and can adapt well to support the delivery of material in diverse ways. Documents and links are closely integrated with their structured domain and there is little opportunity to swap sets of links or different structures on the documents.

Microcosm can also be thought of as a framework for building adaptive hypermedia systems (Hothi & Hall, 1998, Hothi et al., 2000). A linkbase with a built in user model can be used within the filter chain to give different links for different users. The interface components of Microcosm allowed for extensive control to allow applications to place them and manipulate in order to aid in adapting the whole system to a user model.

2.3 Link Services for the World Wide Web

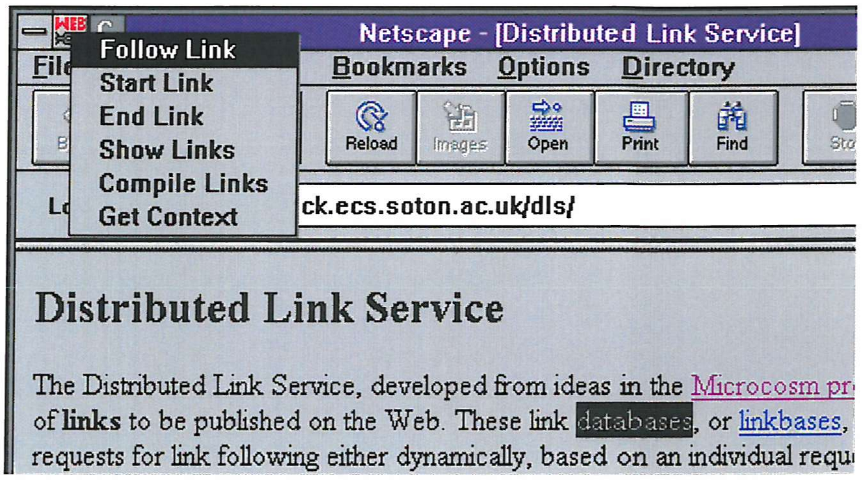
2.3.1 Introduction

As it became clear in the mid-nineties that the World Wide Web would become all conquering many hypertext research groups began to think on how to adapt their systems to integrate with the Web. The simplistic implementation of links in HTML by embedding them in the content was a great step backwards to the hypermedia community but also a contributing factor to the success of the system. It made it very easy for users to create simple collections of interlinked pages without any special environments or tools.

2.3.2 The Distributed Link Service

The group behind Microcosm developed a way to provide a Microcosm-style linkbase service to Web pages in the form of the Distributed Links Service (Carr et al., 1994). The original system was a link server implemented as a set of CGI scripts on a Web server. This could be queried from a Web browser and would return links in much the same way as the linkbase filter in Microcosm.

The client side implementation originally used the 'Light Hypermedia Services' approach developed by Simon Knight (Knight, 1996), described in Section 2.1.5, to add an extra menu to a Web browser. In order to query the link service a user would make a text selection in a Web page and click on the 'Follow Link' menu item, see Figure 2.9. The software would communicate with the browser and cause it to query the DLS server for links using a CGI POST query. The query would include the selected text as well as send some simple contextual information in the form of the URL of the document and the location of the text selection. The server would respond with a page of 'Available Links' in a similar style to the 'Available Links' filter in Microcosm.



Distributed Link Service

The Distributed Link Service, developed from ideas in the Microcosm pr
of **links** to be published on the Web. These link databases, or linkbases,
requests for link following either dynamically, based on an individual requ

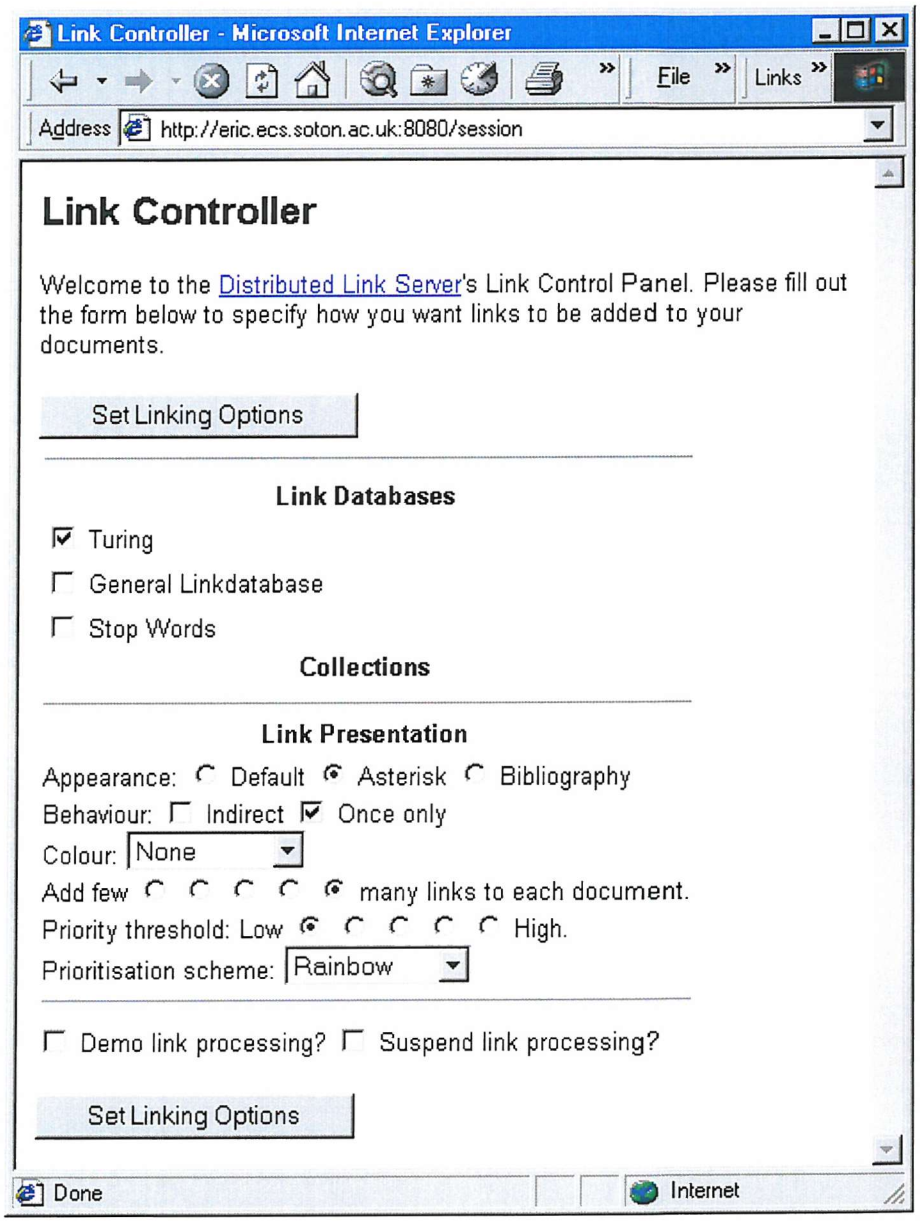
Figure 2.9: Light Hypermedia Services Augmenting a Web Browser.

The 'Show Links' item would send the text selection to the server but it would be separated into individual word and word-pairs which would all be sent to the link server.

The 'Start Link' and 'End Link' menu items allowed users to author links in the same way as was possible in Microcosm. The different types of link were also available and links could be edited through a Form interface. The full generic link functionality opened up a new style of Web publishing. A user could create pages and also author linkbases to match. By making their pages the destinations for generic links they would encourage users to 'come-to' their pages. This is an important concept which we will return to later.

The 'Compile-Links' item caused the DLS to add all possible links to the currently viewed page and return it to the user. This had the same end result as the user selecting the whole document body and clicking on 'Show Links'. Once the DLS was re-implemented as a Web proxy server this became the default behaviour but the history of the system shows that it did not start out as the core functionality. This was simply because it had not occurred to the authors at that time.

The DLS supported the use of multiple linkbases just like Microcosm. An interface Web page, shown in Figure 2.10, allowed the user to specify various preferences and to choose which of the available linkbases they wished to use. This gave the system a simplistic contextual linking capability in that different linkbases could be chosen by user depending on which was most appropriate to the material being viewed. Contextual linking was more strongly supported by the capability to limit the scope of usage of a link or linkbase. A link could be limited to a certain document or to a certain server.


Link Controller - Microsoft Internet Explorer
File
Links
Address
http://eric.ecs.soton.ac.uk:8080/session
Done
Internet

Link Controller

Welcome to the Distributed Link Server's Link Control Panel. Please fill out the form below to specify how you want links to be added to your documents.

Set Linking Options

Link Databases

[✓ Turing]
General Linkdatabase
Stop Words

Collections

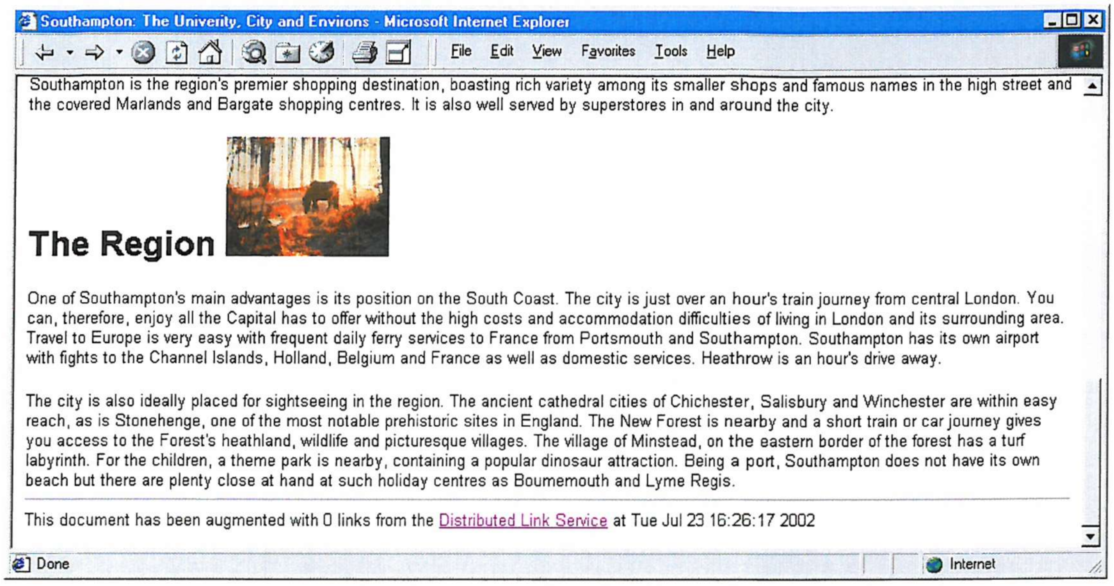
Link Presentation

Appearance: Default [✓ Asterisk] Bibliography
Behaviour: Indirect [✓ Once only]
Colour: None
Add few ○ ○ ○ ○ [✓ ●] many links to each document.
Priority threshold: Low [✓ ●] ○ ○ ○ ○ High.
Prioritisation scheme: Rainbow

Demo link processing? Suspend link processing?

Set Linking Options

Figure 2.10: The Distributed Links Session Control Panel



Southampton: The Univerity, City and Environs - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Southampton is the region's premier shopping destination, boasting rich variety among its smaller shops and famous names in the high street and the covered Marlands and Bargate shopping centres. It is also well served by superstores in and around the city.

The Region

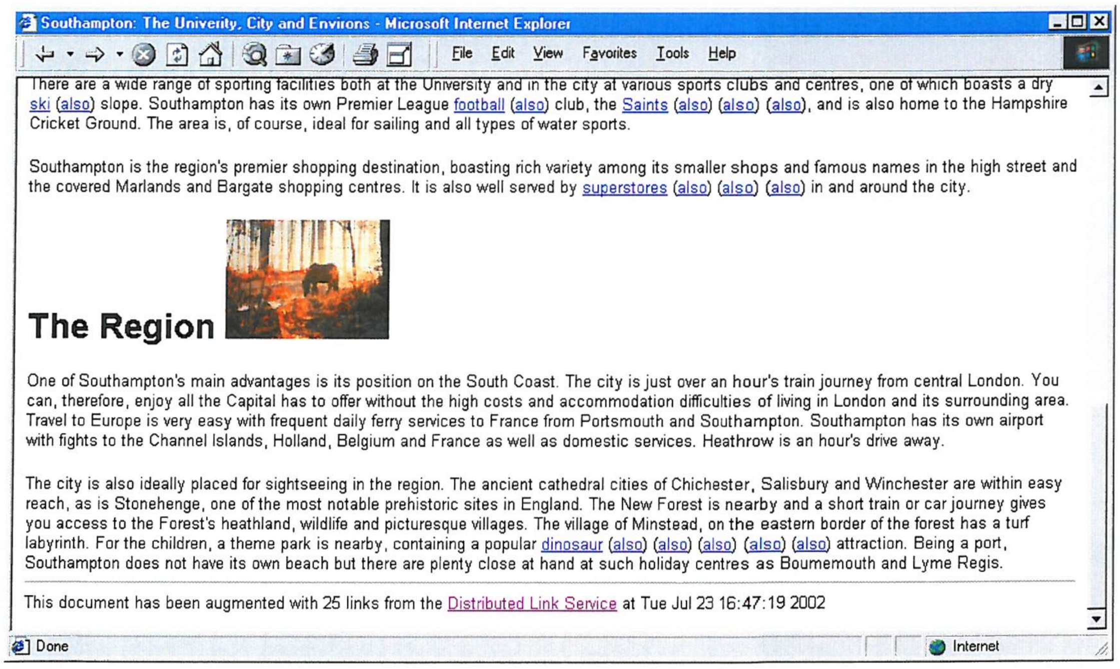
One of Southampton's main advantages is its position on the South Coast. The city is just over an hour's train journey from central London. You can, therefore, enjoy all the Capital has to offer without the high costs and accommodation difficulties of living in London and its surrounding area. Travel to Europe is very easy with frequent daily ferry services to France from Portsmouth and Southampton. Southampton has its own airport with fights to the Channel Islands, Holland, Belgium and France as well as domestic services. Heathrow is an hour's drive away.

The city is also ideally placed for sightseeing in the region. The ancient cathedral cities of Chichester, Salisbury and Winchester are within easy reach, as is Stonehenge, one of the most notable prehistoric sites in England. The New Forest is nearby and a short train or car journey gives you access to the Forest's heathland, wildlife and picturesque villages. The village of Minstead, on the eastern border of the forest has a turf labyrinth. For the children, a theme park is nearby, containing a popular dinosaur attraction. Being a port, Southampton does not have its own beach but there are plenty close at hand at such holiday centres as Bournemouth and Lyme Regis.

This document has been augmented with 0 links from the Distributed Link Service at Tue Jul 23 16:26:17 2002

Done Internet

The normal Web page above is seen below with generic links added by the Distributed Link Service. New links are added on words found in a linkbase about points of interest around Southampton. For instance any occurrence of the word 'Saints' in any document viewed with this browser will receive 4 links related to Southampton Football Club.



Southampton: The Univerity, City and Environs - Microsoft Internet Explorer

File Edit View Favorites Tools Help

There are a wide range of sporting facilities both at the University and in the city at various sports clubs and centres, one of which boasts a dry ski (also) slope. Southampton has its own Premier League football (also) club, the Saints (also) (also) (also), and is also home to the Hampshire Cricket Ground. The area is, of course, ideal for sailing and all types of water sports.

Southampton is the region's premier shopping destination, boasting rich variety among its smaller shops and famous names in the high street and the covered Marlands and Bargate shopping centres. It is also well served by superstores (also) (also) (also) in and around the city.

The Region

One of Southampton's main advantages is its position on the South Coast. The city is just over an hour's train journey from central London. You can, therefore, enjoy all the Capital has to offer without the high costs and accommodation difficulties of living in London and its surrounding area. Travel to Europe is very easy with frequent daily ferry services to France from Portsmouth and Southampton. Southampton has its own airport with fights to the Channel Islands, Holland, Belgium and France as well as domestic services. Heathrow is an hour's drive away.

The city is also ideally placed for sightseeing in the region. The ancient cathedral cities of Chichester, Salisbury and Winchester are within easy reach, as is Stonehenge, one of the most notable prehistoric sites in England. The New Forest is nearby and a short train or car journey gives you access to the Forest's heathland, wildlife and picturesque villages. The village of Minstead, on the eastern border of the forest has a turf labyrinth. For the children, a theme park is nearby, containing a popular dinosaur (also) (also) (also) (also) (also) attraction. Being a port, Southampton does not have its own beach but there are plenty close at hand at such holiday centres as Bournemouth and Lyme Regis.

This document has been augmented with 25 links from the Distributed Link Service at Tue Jul 23 16:47:19 2002

Done Internet

Figure 2.11: The Effect of the Distributed Links Service on a Web Page.

Eventually the proxy implementation was written and adopted. It is shown in use in Figure 2.11. The Netscape menu addition implementation proved very difficult to maintain as each version of the browser exposed differing API functionality. By choosing to implement a proxy the set of user interface implementation difficulties were bypassed.

2.3.3 Web Based Intermediaries (WBI)

When working with the protocols and architecture of the Web the proxy implementation is a powerful way to build more adaptable systems. (Barrett & Maglio, 1998) have created an architecture for building transducers or intermediaries called WBI[1]. This comprises a toolkit of Java building blocks from which it is possible to construct applications. There are 5 basic building blocks which need to be put together in a certain sequence in order to create an application. They are request editors, generators, document editors, monitors and autonomous functions.

- Monitors observe transactions without affecting them.
- Editors modify outgoing requests or incoming documents.
- Generators produce documents in response to requests.
- Autonomous functions run independently of any transaction and perform background tasks.

WBI has been used to build a variety of applications. These include a personalisation proxy for a user to run on their own machine that will annotate pages the user reads. A more powerful system is to use a pair of transducers that can convert HTML or XML into other data formats more suited for transmission using wireless technology. A third use is as a filter for allowing children to use the Web. The system is also used for a transcoding (Barrett & Maglio, 1999, Fox & Brewer, 1996) application, IBM WebSphere Transcoding Publisher. Transcoding is a technology for converting documents between different formats to suit a variety of systems and circumstances such as monochrome displays, low bandwidth connectivity and operating system differences.

2.3.4 Muffin

Muffin (Boyns, 2000) is a framework for writing filters for HTTP messages. The filters work in a chain to modify any aspect of the conversation between a browser and a Web server. The primary aim of Muffin was to provide a way for users to filter Web pages, for instance to remove annoying

[1]Web Based Intermediaries, pronounced "Webby"

banner adverts from Web pages. The system has an architectural similarity to the filter chain of Microcosm in that each filter can alter the message before passing it to the next filter in the system. For this work a new Muffin filter has been written that implements a context-aware DLS. It parses documents sent from a Web server to a browser and adds links into the documents. The key feature is that this DLS filter is implemented to allow the replacement of the way that links are processed and generated. It is this extra layer of abstraction that allows contextual linking to be implemented.

Boyns envisaged four major areas of use for Muffin.

Privacy. Filters were provided to block the transmission and use of cookies by a browser in order for the user to remain anonymous and not have their preferences stored. Other uses included blocking JavaScript code in pages from sending information about a user back to a server.

Security. At the time that Muffin was written Web browsers were more vulnerable to security problems than they are now. Support for Java applets and Microsoft ActiveX controls was in its infancy and such systems were the focus of some speculation over their abilities to allow damage to be done to a computer. Muffin was seen as a way to add further security checks to monitor and control possible harmful effects from such technologies.

Unwanted features. There are many features of Web pages that users find a distraction or irritating. These include the use of banner adverts, Java applets and background music. The system provides filters to remove these and other components from pages. Such abilities have more serious uses when there is a need to adapt Web pages for uses such as viewing on a text only browser or when a browser is limited in capability, such as that on a PDA. Here the system can be used to remove images from the page or other extraneous information to reduce bandwidth required. Web pages can be reformatted to be more suitable to view on a small screen.

Implementing new features. The architecture is an obvious way for developers to implement new functionality. Muffin included several examples including filters for viewing HTTP headers, a way to automatically fill in Web forms and ways to provide fine grained access control to Web sites.

Figures 2.12 and 2.13 depict the flow of events within Muffin for a browser request and the Web server reply.

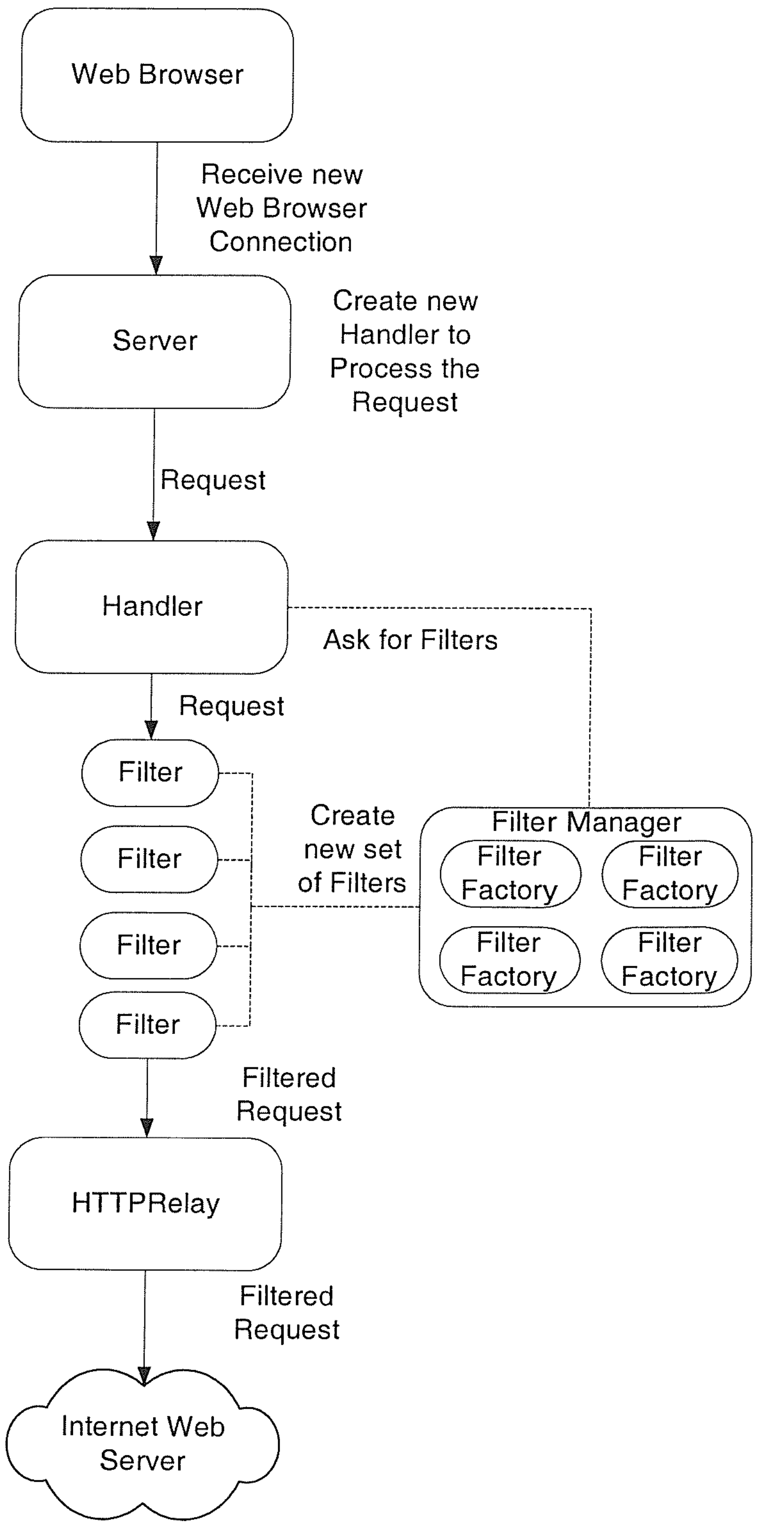


Figure 2.12: Document Request Handling in Muffin (Boyns, 2000).

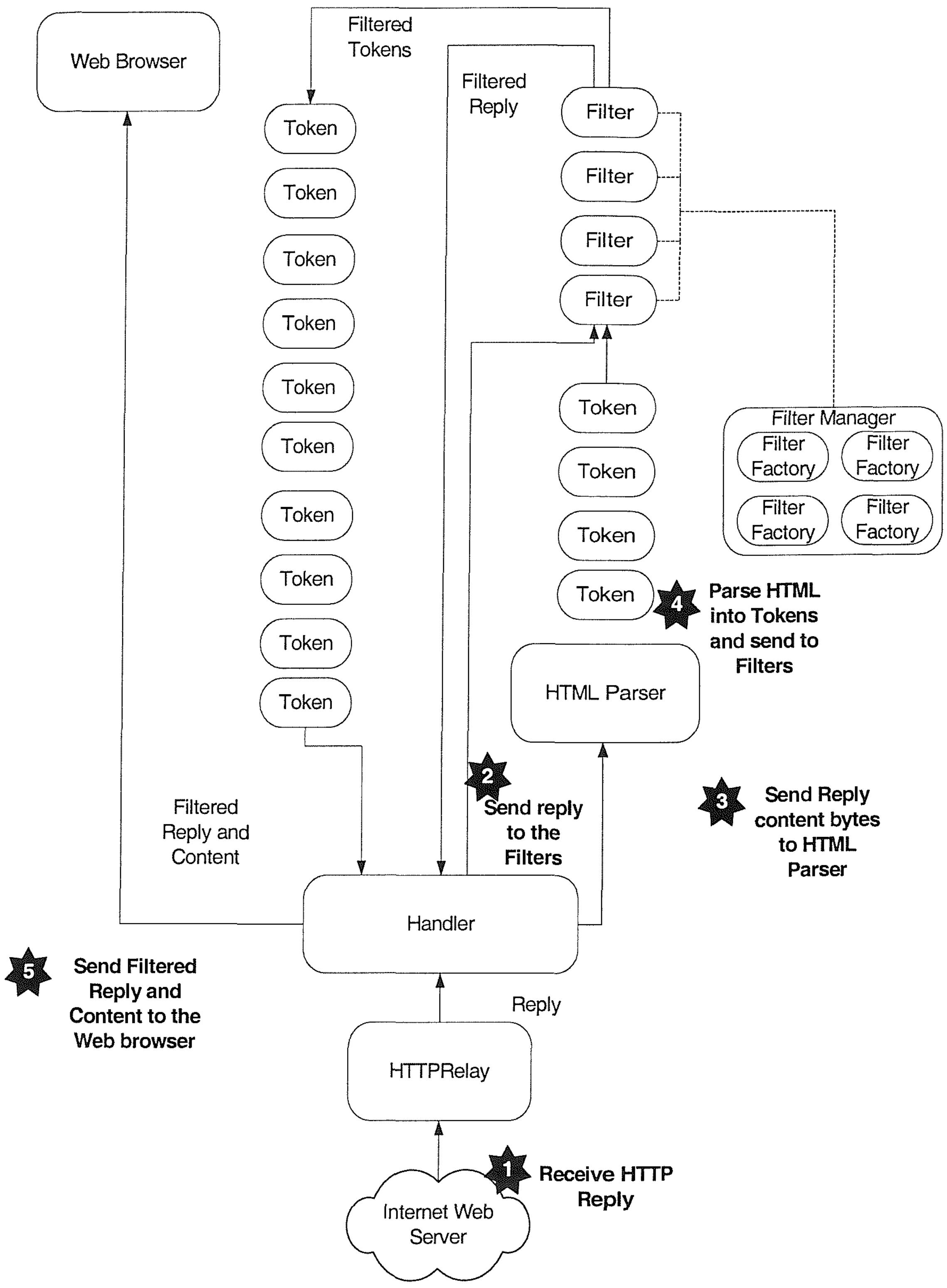
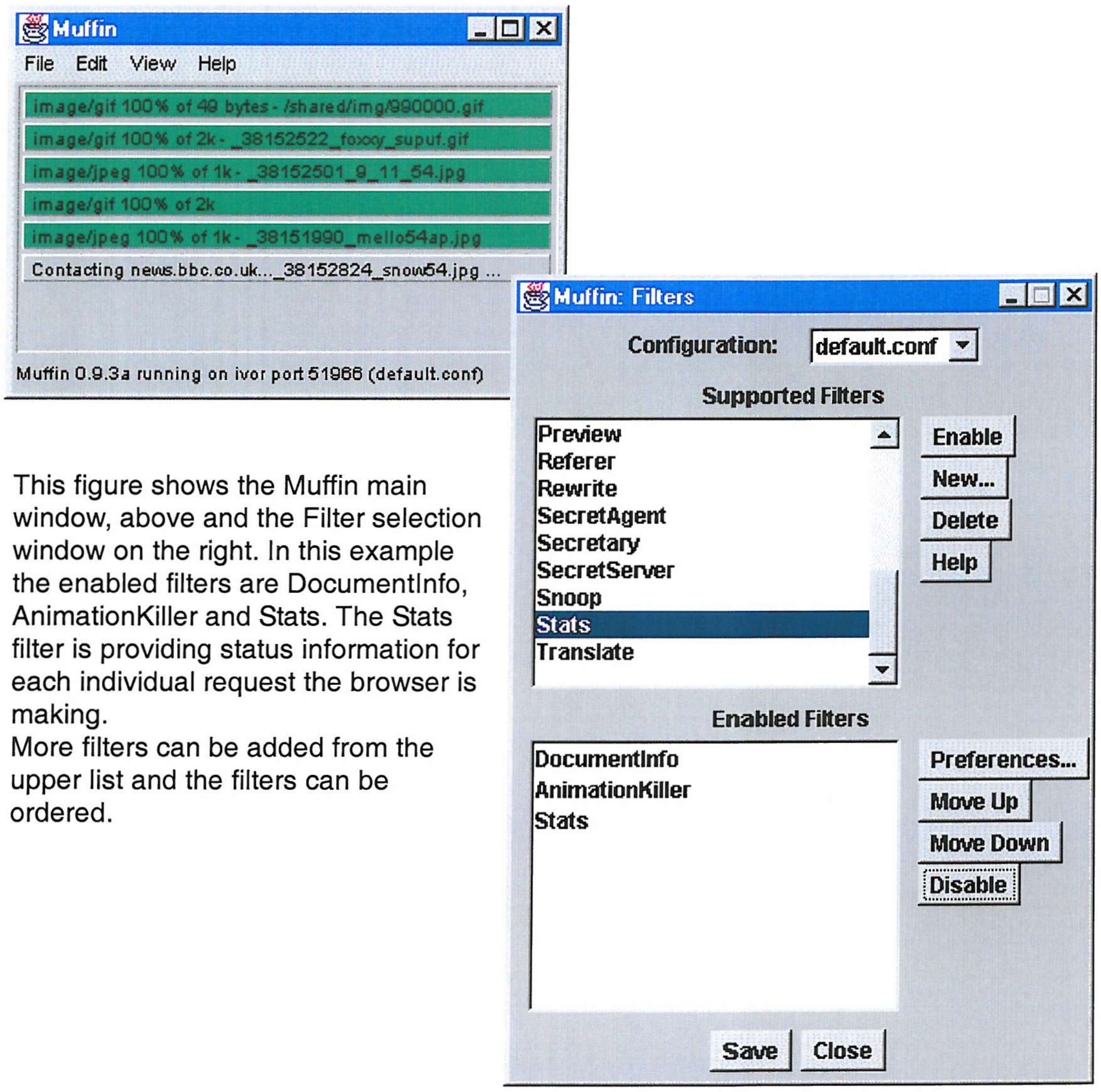


Figure 2.13: Document Reply Handling in Muffin (Boyns, 2000).

Figure 2.14 shows Muffin in use with a number of the default filters. In this case the Document Information, Animation Remover and HTTP Statistics Filters are being used. Their combined functionality is to transform a page and provide information on the requests being made between browser and server. There is considerable similarity to the Filter Manager of Microcosm as shown in Figure 2.2. The important difference is in Muffin each filter alters the raw content of the HTML document before the next filter does the same. In Microcosm the filter chain is responsible for communication between filters but is primarily for the generation and alteration of links.



```
image/gif 100% of 49 bytes - /shared/img/990000.gif
image/gif 100% of 2k - _38152522_foxxy_supuf.gif
image/jpeg 100% of 1k - _38152501_9_11_54.jpg
image/gif 100% of 2k
image/jpeg 100% of 1k - _38151990_mello54ap.jpg
Contacting news.bbc.co.uk..._38152824_snow54.jpg ...
Muffin 0.9.3a running on ivor port 51966 (default.conf)
```

Configuration: default.conf

Supported Filters
Preview
Referer
Rewrite
SecretAgent
Secretary
SecretServer
Snoop
Stats
Translate

Enabled Filters
DocumentInfo
AnimationKiller
Stats

This figure shows the Muffin main window, above and the Filter selection window on the right. In this example the enabled filters are DocumentInfo, AnimationKiller and Stats. The Stats filter is providing status information for each individual request the browser is making.
More filters can be added from the upper list and the filters can be ordered.

Figure 2.14: The Muffin User Interface.

2.3.5 Microsoft Smart Tags

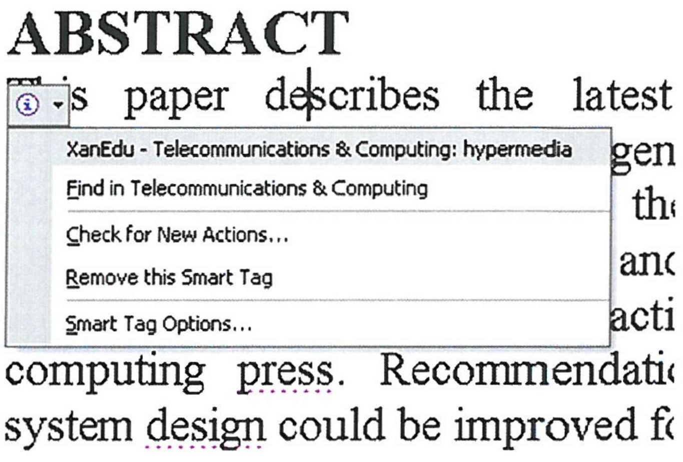
More recently the same basic principle of adding glossary style links, see Section 2.1.5, to Web pages has been taken up by a variety of commercial organisations. Now that Web browsers are vastly more powerful and easier to integrate the task has become a lot simpler. Many products are so much newer than the DLS that their announcement is greeted by the computing press as if new ground is being broken. One of the most recent products was Microsoft Smart Tags (Hughes & Carr, 2002).

For many outside of the Web and hypertext research communities the first time that they were exposed to the idea of generic links was through the announcement that Microsoft planned to add so called 'Smart Tags' to Office XP and the Windows XP version of Internet Explorer 6. As the authors of the operating system and the client applications it would be much easier for them to actually implement the systems properly. However the reaction of various news and review publications was highly negative.

Smart Tags are a facility provided for Microsoft Office applications, which allow software plug-ins to identify regions of a document which are suitable for annotation and to control the processing options available when a user activates (i.e. clicks on) the annotation. Effectively these annotations are synonymous with links.

A Smart Tag consists of two components: a recogniser and an action. The former functions like a simple callback routine, and has a simple Recognise() method which is invoked by the application with a string of text perhaps representing a paragraph, word or cell in the document. The Recognise() method also flags any interesting parts of the text for annotation. The Office application is then responsible for providing the user interface (here a dotted purple underline with a dropdown information menu) for each annotation. The Action object defines the items which can appear in the menu, and controls what happens when any menu item is chosen. The action trivially lists the keywords as menu items, and forms an appropriate URL to trigger the knowledge service when the menu item is selected. See Figure 2.15.

Smart Tags are the basis of a useful implementation of open hypermedia linking. It has been especially designed to allow many recognisers to be active in parallel with the word processing features themselves. It also delivers hypermedia "as you type", as the recognisers are invoked each time a new word has been entered. This is a significant innovation, providing instant feedback to the hypermedia author. However, it is impossible to control the order and timing of the processing of text - in an existing document, paragraphs may only be processed once they are clicked on. Consequently, it is not possible to efficiently establish a document context, and links which depend on certain document features (for example the use of triggering keywords or document structures such as a bibliography) may not be immediately apparent. Lastly, the user's interaction with the annotation and the style of its presentation cannot be controlled.


ABSTRACT

s paper describes the latest gen th anc acti computing press. Recommendati system design could be improved f

XanEdu - Telecommunications & Computing: hypermedia
Find in Telecommunications & Computing
Check for New Actions...
Remove this Smart Tag
Smart Tag Options...

Figure 2.15: Microsoft Smart Tags in use in Microsoft Word.

Smart Tags were first announced as a new feature to be included in the release of Windows XP. The reaction of the computing press was far more passionate than that usually associated with the launch of a hypertext technology. The criticisms[1] were a combination of technical and political and are summarised here.

Parsing and linking of Web pages was to have been enabled by default. All pages would be processed and linked by the system unless the page contained a special META tag in its HTML. Critics argued that this policy should have been implemented in reverse and used it as an example of the company changing the operating system or Web browser without user control[2].

The normal passive experience of using a browser to consume content from a remote server is altered by the inclusion of a link service. The only obvious sign of this to the inexperienced would be the change in appearance of some links in a document. Other link services are more visible to the user as

[1]http://web.archive.org/web/20010710094728/http://public.wsj.com/sn/y/SB991862595554629527.html
http://web.archive.org/web/20010712030846/http://www.zdnet.com/anchordesk/stories/story/0,10738,2771967,00.html
[2]http://news.com.com/2100-1001-267992.html?legacy=cnet

browser plug-ins, proxies or personal agent systems. This informs the user that something more is happening beyond normal Web browsing. When the Smart Tag system is enabled there is no obvious third party involved in the delivery and rendering of the Web page.

The *raison d'être* of a link service is to dynamically enhance a document with links the reader would find useful. The original static text is personalised at read time. There is some irony to this mechanism being cited as the primary offence by the critics. The legal issue was raised that original content could be altered by the browser without permission of the author. This has raised copyright issues over the creation of derivative works[1].

The issue of content being 'surreptitiously' altered is magnified by the crucial factor that the Web is a way of earning a living. It is the only hypertext system that functions as a global marketplace. For instance a review site links readers to affiliate vendors who sell the product under review. The review site receives revenue from such a transaction. If the link service recognises the product and adds its own links to a different vendor then there is potential for lost revenue and the review site is directly damaged[2]. (Neumüller, 2000) has described how keywords are now a commercial commodity to be fought over and the use of keywords out of context is already having an adverse affect on Web sites. The Smart Tag system is open to similar problems, especially given the difficulties of establishing the correct context to link words.

The lack of objectivity in the reviews of this technology is a reaction to a lack of competition. The computer industry is subject to a monopoly in many areas including operating systems, office software and Web browsers. There is a conflict when a company is both content provider and the producer of the means to view the content.

Smart Tags were deactivated in the June 2001 release of Windows XP. Microsoft has stated that it will activate the technology in a future release of Internet Explorer. Smart Tags are implemented in Office XP.

The implementation of Smart Tags does not allow the developer any opportunity to decide how and when to place annotations into documents. There is no scope for contextual linking at the time of parsing the document. When an annotation is activated the system does give the developer a chance to evaluate the circumstances of the link and its context within the document. A handle is given to the

[1]http://www.newsbytes.com/news/01/166676.html
[2] http://www.clickz.com/aff_mkt/aff_mkt/article.php/843801

document which allows the developer to determine the application within which the Smart Tag is working, for example Word, Excel or Internet Explorer, and the rest of the document object model for those applications is available. For instance the Smart Tag application can determine the user, the location of the link and other such document-based data. Therefore it is possible to write Smart Tag applications that have some form of contextual awareness but there is no such facility for determining what links to place where at parse time. It is understood that this restriction is for implementation reasons, the parser is a separate thread within the application and has a lower priority than other threads. There was some difficulty in integrating the parser into the applications. There were possibilities of causing the host applications to freeze up during parsing which were considered unacceptable to the developers. As the system was meant to work in the background and not affect normal user input the decision was made to limit what a developer could do in the API. The other reason given was that parsing could become too expensive an operation for large documents and large numbers of links. A familiar problem encountered by all such systems.

Smart Tags add links as the user types but do not make any attempt to contextualise them. Only when a user activates a link can contextual processing be performed. The resultant application contrasts with my own design.

2.3.6 Other Link Service Products

There have been a number of other such commercial systems producing similar effects to the DLS. Some have not survived the massive shake up of the Internet industry in the year 2000 and some have been acquired by large companies to use in portals and other such systems.

The Flyswat system was a plug-in for Microsoft Internet Explorer that produced a similar effect to the DLS. It underlined words in yellow and gave links to sites. A similar system was TopText, also known as ContextProAdvertising. It was a more aggressively commercial product adding links to commercial sites as a form of advertisement. A third system, EasyLink, gained a little notoriety for the way in which it placed links to a vast array of third party Web sites onto pages.

In most cases these systems are now dead or heavily altered or have little impact on the wider Web community. As with Smart Tags the overwhelming problem is that these systems placed links on Web pages without any consideration of context. Some of the systems aggravated the problem by using too many links and linking to inappropriate sites. As shown with Smart Tags the opposition to these systems has been considerable and organised. The systems have been seen as another way to impose advertising onto Web pages whether that was their prime purpose or not.

Atomica is a similar product but with an emphasis on providing information to employees of an enterprise. The user interface is a simple window which displays search results for any word the user enters. A user can also Alt-Click on any word in any application to activate the system. The company supplies the data sources for the results and does not rely just on public search engines for its data. An enterprise level deployment involves the company integrating the customers existing data sources to provide information to employees.

Another product, RichLink is "the world's first patented, Internet-based service which automatically adds rich contextual information to Web pages which instantly pop up at the user's request." The system has a marked similarity to the DLS and Webcosm in that the company will use the customers own resources to create new links to compile into pages. They will even host the system for the customer. It is difficult to gauge exactly how it is uniquely different from the other systems listed here without contacting a sales representative.

2.4 Summary

Link services are a component of open hypermedia services which have been used to provide linking to systems, either integrated in systems or as stand alone products. This culminated with the Open Hypermedia Systems workshops and the development of generalised models of linking to allow for interoperability between open hypermedia systems.

To improve the quality of linking is the goal of adaptive hypermedia, here systems use models of users, documents and goals to aid in providing links that are better suited to users. The systems have mainly been used in educational applications whereby documents and navigation links are altered to reflect the users stage of learning.

The advent of the Web has provided new opportunities for link services to improve the rudimentary hypertext features of a browser. Systems such as the DLS from Southampton add links to web pages on demand via implementations such as taking the form of a web proxy. These systems have tended to use a single model of a link and a corresponding single way to compute which links to use and when to use them in a document. Later proxy-based application development systems such as WBI and Muffin have provided methods to add to these functionalities though no context aware linking applications had been produced using them until now.

2.5 Conclusion

This chapter has presented a brief history of open hypermedia with particular attention to the link service component of such systems. The use of applications such as the DLS to improve the linking facilities of the World Wide Web has been examined and the success or otherwise of various commercial systems that have attempted to do so. The goal has been to find hypermedia research areas which can provide pointers on to how to design a contextual link service. The adaptive hypermedia community has many years of experience in techniques to provide more useful links for users in particular situations. The core of these systems is sophisticated modelling of users and the goals of the system.

The next chapter describes the use of a link service for the Web with a document management system. The hard lessons learnt from trying to integrate the service with an operational system using real documents and real data sources produced many new issues that needed addressing. First attempts were made to serve links in context as required to be useful to the service and users. This led to the conclusion that a new design of link service would be needed in order to be able to provide contextual links. Chapter 4 looks closely at what exactly context is and Chapter 5 presents the architecture of a contextual link service.

3 Integrating Link Services With A Document Management System

3.1 Introduction

An open hypermedia system is always championed as a way to improve documents and information. It adds knowledge to a corpus of information and aids users who want to find something. There are known problems and design issues with the implementation of a system such as the DLS which need to be addressed. One of the ways of improving its effectiveness is to create tighter integration with the system producing the documents. Whilst being an entirely independent system has strong advantages there reaches a point when it prevents innovation and experimentation. In this chapter the integration issue is explored, in particular the effects that can be achieved when the link service and document provision service are more closely related.

The experiments take the form of integrating a Web based document management system with link services in order to enhance the documents. The AIMS (Academic Information Management System) document management system described in this chapter provided a web based service to staff of the author's Department allowing them to upload and view administration documents. A number of experimental combinations of link service and document management system are described aiming to find new ways to tailor links to users and to the documents themselves. The chapter explores the limitations of using current link service technology and builds a set of requirements for a new link service that can provide a greater range of adapted links.

Effective adaption requires better understanding of the documents, the users and the circumstances in which the links were being used. The work leads to the realisation that no single link service model could do all of the possible things expected of such a system and that a new link service is required that is completely open to new linking methods and models. The results of the work described here are a requirements list for a new design of contextual link service.

It was the conclusions drawn from these experiments and a deeper look into the meaning and uses of context (Chapter 4) that led directly to the design of the contexual link service presented in Chapter 5.

3.2 The AIMS Document Management System

3.2.1 Background

The Electronics and Computer Science Department had a considerable need for an electronic document management solution. Almost no administrative information was available online. There was no policy for keeping paper documents and no electronic archive. Documents were produced by secretaries, printed and sometimes the source files lost. For important committees the minutes and associated reports were printed, photocopied and distributed by hand to academic staff, a process taking many hours. Due to the confidential nature of the material it was felt that the internal post system was not secure enough to be used. The use of electronic distribution was certainly not a possibility. Many of the academics preferred to have paper copies of the documents and a significant number did not possess the technical skills to print an email attachment.

No central archive of the source files was kept and in many cases the electronic files were deleted by the secretary from their own machine. They would use the file of the previous edition as the starting point for the new, but not keep a backup. The situation was exacerbated by a lack of training for administrative staff. Because none of the administrative staff had any knowledge of creating Web documents they could not and would not place their documents on the Department site and would never have been instructed to do so. The only information online was either provided by a select few Web-enlightened academic staff or by Webmasters under the instruction of senior Department staff.

For a Department featuring a world class research group specializing in hypermedia and Web technologies here was a challenge that needed meeting. It was also felt necessary that we should 'put our own house in order' if we were to extol the virtues of hypertext and linking technologies to the wider world.

The overall goal of the AIMS project was to deliver a document management system with a Web interface. This system would enable the authors of documents to directly contribute their work to a Web site without needing to know how to create HTML, without needing to know the mechanics of where their files would need to be located and without the direct intervention of anyone else in the process. The system could not cause a large upheaval in working practice or the employment of a full time administrator. The system would become an electronic archive and a resource for enabling document reuse. A long term aim was to reduce printing costs by distributing documents electronically and allow each staff member to decide whether to print them or have them printed for them. It should be noted that at the start of this project, January 1997, the electronic skills of many members of the Department, both academic and administrative, were very limited. Many were not users of the Web. The author also felt that he could not impose too many changes on working practice as the natural inertia of the Department would be considerable.

The project did not set out to create a paperless environment. The administrative system was geared towards creating printed documents. Archiving and publishing them on AIMS would be of secondary importance. With this in mind the system was designed to allow people to work as they always had done using the tools that they wished to use. No software standards or working practice rules could be imposed because the author was in no position to do so. For example it was not possible to ask everyone to use a certain brand of word processor in order to exploit its features as part of the system design. Approximately half of the Department staff use a variety of Unix and half use Microsoft Windows. With many commercial document management systems available the user is expected to be able to view the stored file on their local machine. To be more explicit there is an assumption that the user has Microsoft Word for Windows and can view the file. Even today this expectation is not valid in academia, especially in a Computer Science Department. In industry, where Microsoft has a near monopoly on normal workstations the assumption has more validity. Many other document management systems just act as a file store and expect the user to download the file and read it locally. AIMS explicitly does not make this assumption.

This contrasts with an industrial environment where it is normal for the organisation to impose a choice of software or a style of working on a workforce for financial reasons. Here in a University Computer Science Department it is almost impossible to tell people how to work on a day to day basis, especially when it comes to software choices. The design of AIMS reflects this ethos (some might say chaos). For instance the minimum of information is required about each document in order not to discourage users from using the submission form. Users are not expected to enter large amounts of metadata in order to classify their own work. The site was also designed for access by any Web browser rather than just the most modern. Features such as frames, JavaScript, Java applets and advanced HTML tags were not used. It must be remembered that in 1997 Netscape 2 was not formally released and Internet Explorer had yet to be successful. Another important factor is that the systems used by administrative staff tend to be the oldest and of poorest specification in the Department.

3.2.2 System Summary

The AIMS system is an automatically generated document database accessible from any Web browser. It is implemented as a Lotus Notes database application and hosted by a Lotus Domino Web server. Submission of documents to AIMS is achieved by a user completing a Web form and submitting a file produced by their word processor application. The AIMS application imports the file into the database and the Domino Web server renders the content of the file as a Web page automatically. What this means to the user is that if a Word for Windows file is submitted into AIMS the server will instantly generate a Web page of the content of that file. The page also allows the user to download the original file, acting as a file store, and the page becomes part of the full text search index.

3.2.3 An Introduction to Lotus Notes

Lotus Notes is an application development environment for large, groupware oriented, distributed applications. The features are numerous and include a powerful underlying object oriented database, programming languages for creating bespoke applications and a range of rich-text import libraries. These are crucial as they allow the system to import the content of word processed files that users submit to the system. The name 'Lotus Notes' refers to the overall product family and also to the client application portions of the product. 'Lotus Domino' is the name of the server product line that hosts 'Lotus Notes' databases. Originally users ran the Notes client to access database applications and their email. In 1997 Lotus added Web server functionality to the product allowing Web browsers to access the system though with a limited capability. The functionality improved during the lifetime of the project. The Domino server generates Web pages on demand for all aspects of the design.

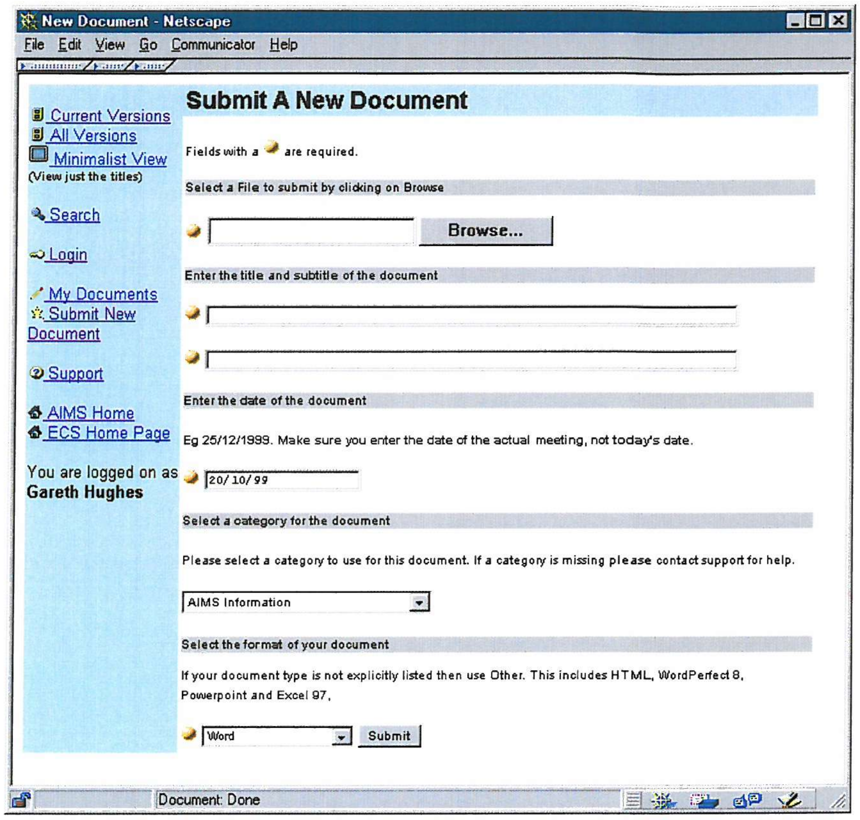
In Lotus Notes a database is filled with Documents created by the completion of Forms. Forms describe the layout of Fields of data. Fields contain individual items of data ranging from a simple number to a rich text field capable of holding many megabytes of varied information. Documents in the database are listed by Views. Views are also designed to show certain subsets of the data in the database. When a user requests an AIMS document from their browser the server places data into the fields according to the chosen form design. The result is converted to HTML by the HTTP server component of Domino. The main project deliverable was a Lotus Notes database template file. This is used to create new AIMS databases on a server. For instance there are currently 5 AIMS databases in use on the project server. If the template is updated the databases will be automatically upgraded.

There is wide confusion over exactly what Lotus Notes actually is. Lotus Notes is not a document management system and it is not just an email system though that is a key component. It is a complex and adaptable starting point for organisations to create applications that reflect the way a company is organised and works. A large investment is required to create the bespoke applications that are its speciality. This is why major corporations use and benefit from it most and may explain why it is little used in academia. The key ability to import various word processing formats and then generate Web pages of the content on the fly was found to be unique at the time. The alternative to using Lotus Notes for the AIMS project would have been to develop a bespoke system.

3.2.4 Features of AIMS

The following sub-section describes the features for users of the AIMS system. It is a description of the fourth version of the system finished at the end of the 3 year project.

Submission Process. The process for a user to submit a file to AIMS is simplified to encourage usage. The user writes a document using a Word processing package. Once the document is finished it will usually be printed and distributed. The user would also then submit this file to the AIMS system by completing the Web form shown in Figure 3.1.



You are logged on as
Gareth Hughes

Submit A New Document

Fields with a ● are required.

Select a File to submit by clicking on Browse

Enter the title and subtitle of the document

Enter the date of the document

Eg 25/12/1999. Make sure you enter the date of the actual meeting, not today's date.

20/10/99

Select a category for the document

Please select a category to use for this document. If a category is missing please contact support for help.

AIMS Information

Select the format of your document

If your document type is not explicitly listed then use Other. This includes HTML, WordPerfect 8, Powerpoint and Excel 97,

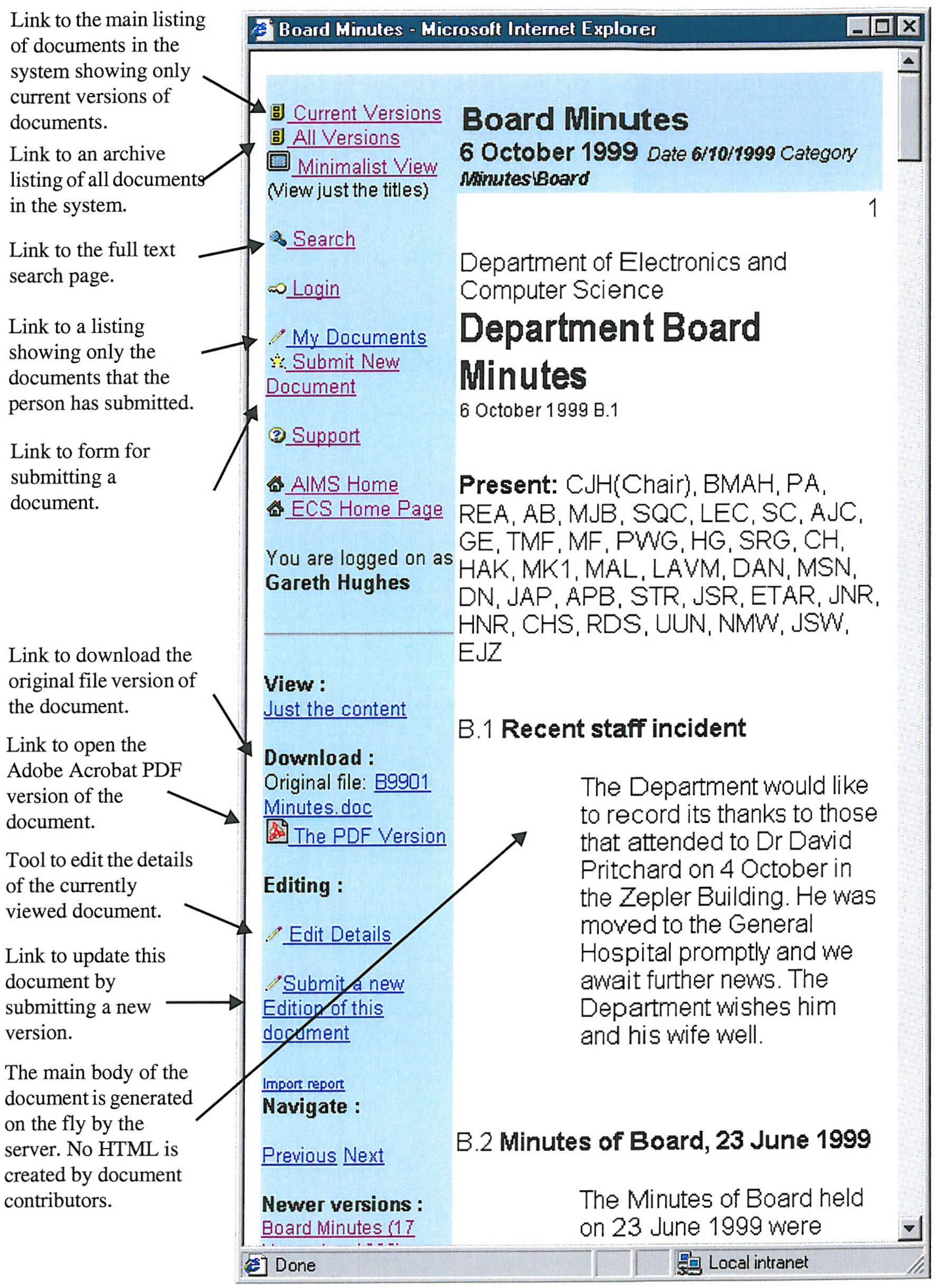
Word

Figure 3.1: The AIMS Document Submission Web Form.

The person selects the actual file to put into the system using the Browse button. They then enter in a Title, Subtitle, document or meeting Date and choose a Category from the pre-determined list of categories. When the user presses the Submit button the browser sends the file and form data to the server. At this point a complex process occurs to create a new document in the Notes database, populate it with the metadata and spawn off the file conversion and import process. The result is a Web page of the document the user submitted.

When this method was designed the ability to upload files from a Web browser was a new feature and the method a novel one.

Browsing Features. Figure 3.2 below illustrates a page from AIMS and highlights some of the tools and properties of the system.



Link to the main listing of documents in the system showing only current versions of documents.

Link to an archive listing of all documents in the system.

Link to the full text search page.

Link to a listing showing only the documents that the person has submitted.

Link to form for submitting a document.

Link to download the original file version of the document.

Link to open the Adobe Acrobat PDF version of the document.

Tool to edit the details of the currently viewed document.

Link to update this document by submitting a new version.

The main body of the document is generated on the fly by the server. No HTML is created by document contributors.

Board Minutes - Microsoft Internet Explorer

Current Versions
All Versions
Minimalist View
(View just the titles)

Search

Login

My Documents
Submit New Document

Support

AIMS Home
ECS Home Page

You are logged on as **Gareth Hughes**

View :
Just the content

Download :
Original file: B9901 Minutes.doc
The PDF Version

Editing :

Edit Details

Submit a new Edition of this document

Import report
Navigate :

Previous Next

Newer versions :
Board Minutes (17

Board Minutes

6 October 1999 *Date **6/10/1999** Category **Minutes\Board***

1

Department of Electronics and Computer Science

Department Board Minutes

6 October 1999 B.1

Present: CJH(Chair), BMAH, PA, REA, AB, MJB, SQC, LEC, SC, AJC, GE, TMF, MF, PWG, HG, SRG, CH, HAK, MK1, MAL, LAVM, DAN, MSN, DN, JAP, APB, STR, JSR, ETAR, JNR, HNR, CHS, RDS, UUN, NMW, JSW, EJZ

B.1 **Recent staff incident**

The Department would like to record its thanks to those that attended to Dr David Pritchard on 4 October in the Zepler Building. He was moved to the General Hospital promptly and we await further news. The Department wishes him and his wife well.

B.2 **Minutes of Board, 23 June 1999**

The Minutes of Board held on 23 June 1999 were

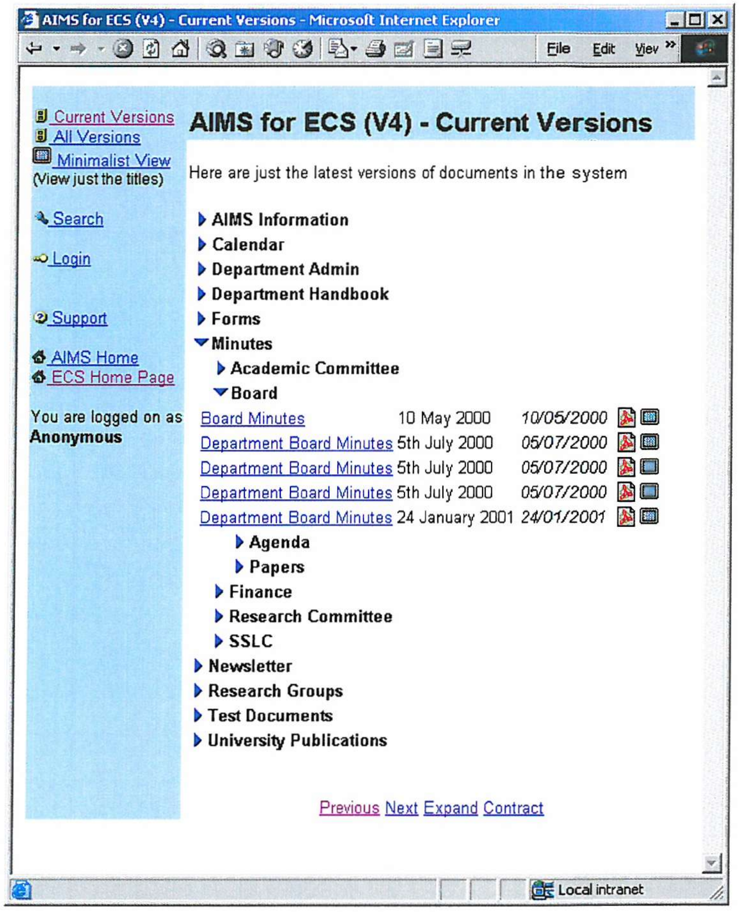
Done

Local intranet

Figure 3.2: Features in AIMS.

Some of the main design features of AIMS are listed below.

Access to documents via generated listings. The documents in the AIMS system are listed by automatically generated listings. A number of listings are available, the main listing contains only current document versions. Older versions of documents can be accessed from a Document Archive listing. Figure 3.3 shows one of the listings views.


AIMS for ECS (V4) - Current Versions - Microsoft Internet Explorer

File Edit View

Current Versions
All Versions
Minimalist View
(View just the titles)

Search

Login

Support

AIMS Home
ECS Home Page

You are logged on as
Anonymous

AIMS for ECS (V4) - Current Versions

Here are just the latest versions of documents in the system

- **AIMS Information**
- **Calendar**
- **Department Admin**
- **Department Handbook**
- **Forms**
- **Minutes**
  - **Academic Committee**
  - **Board**

Board Minutes	10 May 2000	*10/05/2000*	
Department Board Minutes	5th July 2000	*05/07/2000*	
Department Board Minutes	5th July 2000	*05/07/2000*	
Department Board Minutes	5th July 2000	*05/07/2000*	
Department Board Minutes	24 January 2001	*24/01/2001*	

  - **Agenda**
  - **Papers**
  - **Finance**
  - **Research Committee**
  - **SSLC**
- **Newsletter**
- **Research Groups**
- **Test Documents**
- **University Publications**

Previous Next Expand Contract

Local intranet

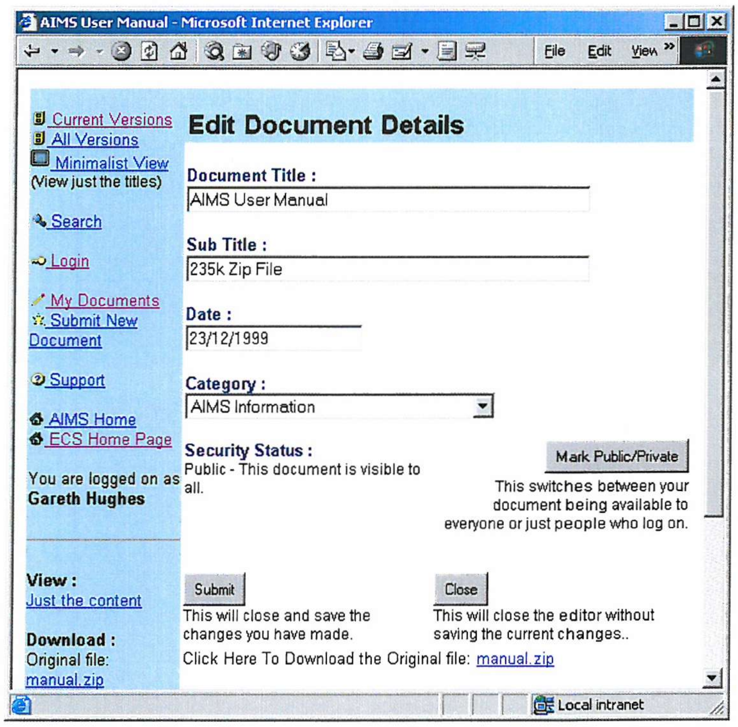
Figure 3.3: A Document Listing View of AIMS.

Version control facilities. A form of version control was designed for the later versions of AIMS. An important characteristic of the system is that each document had a unique number. URLs to documents in AIMS used this unique number. The version control facilities were designed with the assumption that it was more important for bookmarks to documents in AIMS to stay constant rather than break them. If a document was updated with a newer version it assumed the unique number of the existing document so that a bookmark to that document would always retrieve the latest version.

The older version would be given a new unique number. The justification being that for the type of information stored in AIMS the user would usually want to see the latest rather than earlier versions. All versions of documents were available through the Archive listing of the database.

Automatic creation of Adobe Acrobat PDF files. One of the additional features of AIMS is an ability to create an Adobe Acrobat PDF file of documents submitted to the system. This has been a popular facility especially when some documents are not represented well as a Web page or are designed to be printed. For example a popular PDF download from the site is the Travel Expenses Claim form for the Department.

Details of documents can be edited online. Once a document is in the system it is possible for the contributor to alter the details of that document via an online form shown in Figure 3.4. This is useful for correcting spelling errors in fields, changing the security settings of the document or changing the category of a document.



Edit Document Details

Document Title : AIMS User Manual

Sub Title : 235k Zip File

Date : 23/12/1999

Category : AIMS Information

Security Status : Public - This document is visible to all.

This switches between your document being available to everyone or just people who log on.

You are logged on as **Gareth Hughes**

This will close and save the changes you have made.

This will close the editor without saving the current changes..

Click Here To Download the Original file: manual.zip

View : Just the content

Download : Original file: manual.zip

Figure 3.4: The Document Editing Form in AIMS.

Full Text Search Engine. Other interface features include a full text search page. Full text searching is built into Lotus Notes. A page was built to tailor searching to the AIMS design.

3.3 Experimental Integration of AIMS with a Link Service

3.3.1 Basic Link Service Integration

As well as providing a service to the Department the AIMS system was created to be used as a testbed for experiments in integrating DLS-style link servers. As a consequence of their automated creation the pages produced by the AIMS site had no actual links within the body of the pages. However the documents were highly related in nature so plenty of opportunities were available to link topics and keywords in the documents to each other and to the Department's main Web site. The first steps were to use the DLS in the normal way as a Web proxy supplying linkbases of generic links.

A company was started to commercialise Microcosm. In 1997 it also took on the DLS and created a product version called Webcosm. The core technologies were to appear in all of the products from the company, now called Active Navigation Ltd. In the early experimental work described in Chapter 3 both the DLS and Webcosm have been used, at times interchangeably. From the point of view of the work described here they are interchangeable in principle but one or the other has been chosen for a particular strength of its implementation.

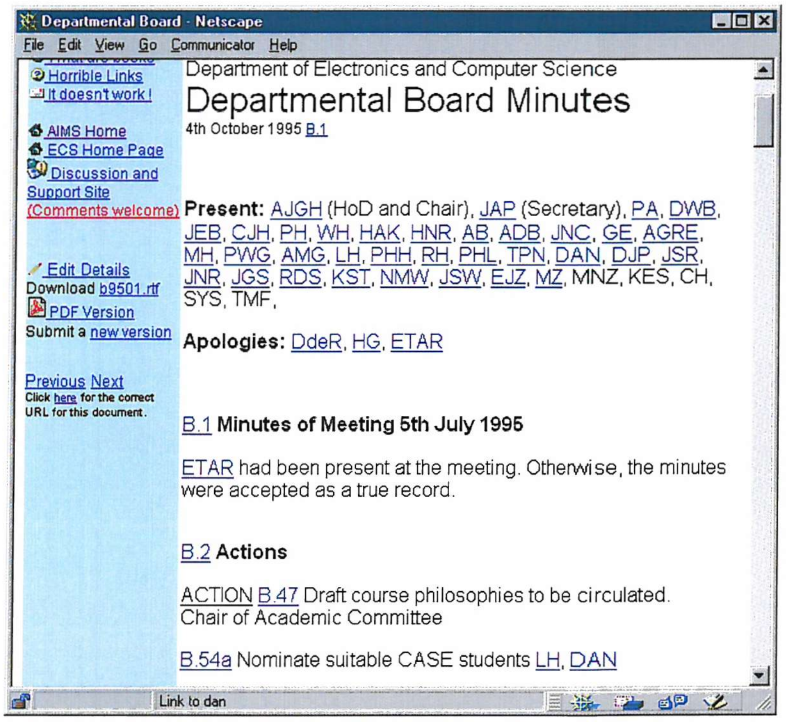
A number of linkbases were created with relevance to the material in AIMS or to pages on the Department Web site. The first consisted of a generic link for each undergraduate course taught by the Department to the relevant course home page on the Department Web site. For example 'CM142' is the code for the first year computer science course 'Advanced Programming'. This linkbase was generated by parsing the existing Department Web page listing all of the Undergraduate courses.

A large linkbase linking user id's and user names to people's home pages was generated by writing a script to access the Department personnel database. For each person in the Department three links were generated. One on user name, full name and initials followed by surname. E.g. 'gvh', 'Gareth Hughes' and 'G.V. Hughes'. For each link the destination was to their official home page.

These two simple examples of linkbases highlight many common features. They illustrate many of the positive contributions that link services make. The creation and maintenance of the two linkbases is a simple task as each is generated from a script. These links are highly applicable and will be useful in many documents. The number of links that will appear in documents against the effort to make them is highly favourable. The linking of people's names in order to allow a user to find out more about the person is not so useful or easy to do as normal generic links on acronyms. The first problem is that if the linkbase is generated from a data source such as the user accounts system then only current members of the Department will be linked. Those who have left will no longer be linked. In a system which is essentially a history of the formal workings of the Department this is not so useful.

Another, much harder to solve, problem is that of understanding when to place a link in a document. A classic, real-life, example is that the example given includes a link on a person who's username is 'AND'. The link service does not know when to place the link and when not to. It has no idea if the characters 'and' refer to the person or just to the word 'and'. Even a cursory examination of the problem indicates that this would be very hard to solve.

Both problems can be seen manifesting themselves in Figure 3.5. It shows a document of meeting minutes being linked by the DLS using the username linkbase. People who have since left the Department are not linked and 'AND' is linked inappropriately.



Department of Electronics and Computer Science

Departmental Board Minutes

4th October 1995 B.1

Present: AJGH (HoD and Chair), JAP (Secretary), PA, DWB, JEB, CJH, PH, WH, HAK, HNR, AB, ADB, JNC, GE, AGRE, MH, PWG, AMG, LH, PHH, RH, PHL, TPN, DAN, DJP, JSR, JNR, JGS, RDS, KST, NMW, JSW, EJZ, MZ, MNZ, KES, CH, SYS, TMF,

Apologies: DdeR, HG, ETAR

B.1 **Minutes of Meeting 5th July 1995**

ETAR had been present at the meeting. Otherwise, the minutes were accepted as a true record.

B.2 **Actions**

ACTION B.47 Draft course philosophies to be circulated. Chair of Academic Committee

B.54a Nominate suitable CASE students LH, DAN

Figure 3.5: Minutes of a Meeting Augmented with Standard Generic Links from the DLS.

The third linkbase included listings of more generic terms and acronyms such as research group acronyms and other abbreviations such as 'HoD' for 'Head of Department'. These definitions were gathered and written manually into a linkbase by the author.

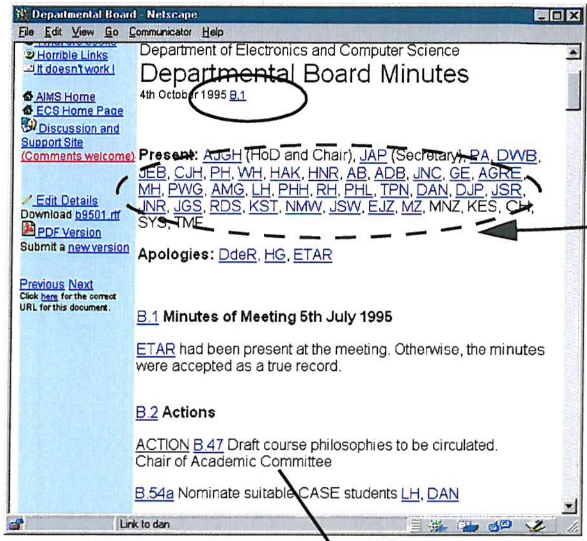
These linkbases and techniques for creating them were the traditional methods in use for a number of years in a variety of projects relating to Microcosm and the DLS. Either a linkbase was written manually or simple scripts would be written to harvest existing information from a source or Web site.

More advanced linking techniques can be employed depending on the level of programmatic access to the document management system and the link service. The following sections demonstrate this showing the use of pattern matching linking, the effect of integrating the link service directly into the document management system, the effects of pre-compiling static links as opposed to real-time generation and the use of document dates as an aid to more relevant linking. Each experiment demonstrates limitations of the current technologies and provides requirements for the design of a new link service.

3.3.2 Using Pattern Matching Techniques for Creating Link Anchors

One of the major types of documents stored in AIMS were minutes of meetings. These provide an opportunity to go beyond simple linking on keywords as the documents are related to each other. Frequently items in minutes will be following on from Actions and Items in previous editions of the minutes. The plan was to use a link service to automatically link these relationships together. The minutes of the Department Board of Electronics and Computer Science contain item numbers of the form 'B.n' where 'n' is a number. The Distributed Links Service was used to link together these item numbers with the goal being that readers could follow the trails of issues back through time. Figure 3.6 illustrates the process.

Item B.2 contains a reference to an ACTION B.47 in the previous set of minutes. The user follows the link B.47. This link triggers an script on the AIMS server.



Departmental Board - Netscape

Horrible Links
It doesn't work !
AIMS Home
ECS Home Page
Discussion and Support Site
(Comments welcome)
Edit Details
Download b9501.rtf
PDF Version
Submit a new version
Previous Next
Click here for the correct URL for this document.

Department of Electronics and Computer Science

Departmental Board Minutes

4th October 1995 B.1

Present: AJGH (HoD and Chair), JAP (Secretary), PA, DWB, JEB, CJH, PH, WH, HAK, HNR, AB, ADB, JNC, GE, AGRE, MH, PWG, AMG, LH, PHH, RH, PHL, TPN, DAN, DJP, JSR, JNR, JGS, RDS, KST, NMW, JSW, EJZ, MZ, MNZ, KES, CH, SYS, TME

Apologies: DdeR, HG, ETAR

B.1 **Minutes of Meeting 5th July 1995**

ETAR had been present at the meeting. Otherwise, the minutes were accepted as a true record.

B.2 **Actions**

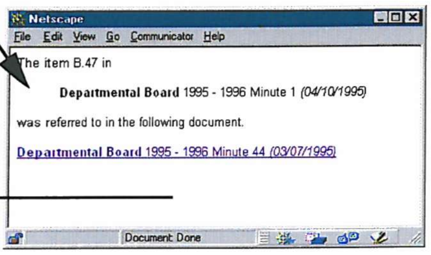
ACTION B.47 Draft course philosophies to be circulated. Chair of Academic Committee

B.54a Nominate suitable CASE students LH, DAN

Link to dan

Another set of links in the Link Service provides a link from the initials of members of the Department to their home page. These links are generated based on the information stored in the Department's personnel database.

The script searches the database to find the most likely document containing the correct item B.47 by using the date of the document the user is viewing.



Netscape

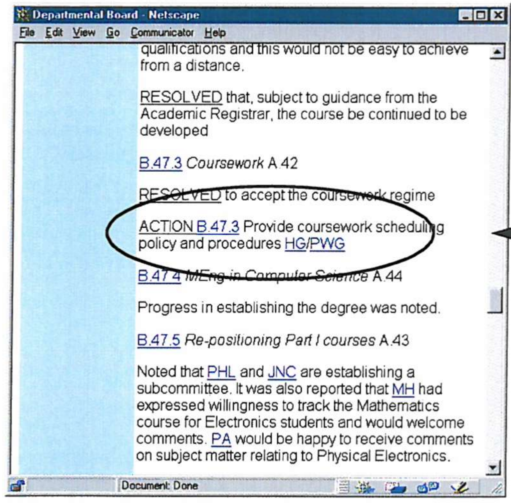
The item B.47 in

Departmental Board 1995 - 1996 Minute 1 *(04/10/1995)*

was referred to in the following document.

Departmental Board 1995 - 1996 Minute 44 *(03/07/1995)*

Document: Done



Departmental Board - Netscape

qualifications and this would not be easy to achieve from a distance.

RESOLVED that, subject to guidance from the Academic Registrar, the course be continued to be developed

B.47.3 *Coursework* A.42

RESOLVED to accept the coursework regime

ACTION B.47.3 Provide coursework scheduling policy and procedures HG/PWG

B.47.4 *MEng in Computer Science* A.44

Progress in establishing the degree was noted.

B.47.5 *Re-positioning Part I courses* A.43

Noted that PHL and JNC are establishing a subcommittee. It was also reported that MH had expressed willingness to track the Mathematics course for Electronics students and would welcome comments. PA would be happy to receive comments on subject matter relating to Physical Electronics.

Document: Done

The script returns a link to the referred document. In the body of this document is the Action B.47.3

Figure 3.6: Using Pattern Matching Techniques in the DLS to Link Issues to Items in Minutes.

The application worked only partially, it would often fail to match correctly on words and subsequently fail to add links where there should have been links. When the document parsing worked the application would correctly link documents together.

The problem with this application was that it only applied to a few documents in the system matching a very specific pattern. The implementation of the application made it difficult to present easily understood results to the user when a link was followed. Hence it was not obvious how to use the links provided and navigate back through the documents. Because of inaccuracies and variations in the patterns of codes used in different documents the pattern matching was seldom successful.

The conclusion drawn from the experiment was that pattern matching can work but with a considerable maintenance cost and a large failure rate for matching. A major problem was that there would need to be large number of patterns to match all the different types of documents in the system. These would all need maintaining to cope with users changing their styles of working. For instance some types of minutes use 'B.1' and some use 'K-3a' etc. This leads to a second problem, if a full implementation was contemplated the first problem to be overcome would be to find a way for the link service to switch between patterns and, more importantly but much harder to implement, it needs to know when to do it.

This is an example of a common problem in attempting to work with free text. Authors of documents are rarely accurate or consistent. If the content is difficult for the developer to work with then the logical solution is to attempt to place links based on document structure. For example a program could look for text with the style 'Heading 1'. This seems more natural as XML (W3C Consortium, 1998a) has taken hold and structured text has become more widespread but unfortunately the technique is completely unworkable with the real documents held in AIMS. The surprising reality is that none of the documents in AIMS have been authored using even simple styling or heading names. It is not known how much training in using Word processors the staff have been given but it would seem reasonable to assume that they know how to use styles. Hence it was a surprise to the author to find no structured markup to work with.

Therefore whilst many great things are possible with structured text, demonstrated by the explosion in popularity of XML, none of them can be done here because structured text is not available in this real life situation. The author has also had some experience of other industrial situations where even simple headings and other style information or metadata is not used by people creating word processed documents. Perhaps it is an indication that there is a lag between the work done by computer scientists familiar with new technologies and the 'real world'.

The techniques briefly examined here stem from work done elsewhere in the group on linking in scientific journals. Citation linking (Garfield, 1979) is the automatic linking of references in research papers. It is one of the major areas which have fuelled the development of the link service within IAM and elsewhere. The major one being the Open Journals project (Hitchcock et al., 1997, Hitchcock et al., 1998) in which a system to link references between journals in a number of electronic archives was achieved by the creation of a citation agent. This version of the DLS could recognise references in a paper, match the citations against a database of abstracts and add links to the database entry where one was found. The system was designed to load new patterns and methods of recognition allowing the authors to refine their techniques to match increasing numbers of references. A name recognition version was also written matching the combination of people's surnames and initials. The project had higher quality resources to work with in the form of scientific papers rather than simple word processed documents. These are more structured and the formatting is more consistent.

Citation indexing and linking is now a powerful resource for researchers. For instance this thesis was written using such a system as the major source of references. The Research Index or CiteSeer (Bollacker et al., 1998) system is a digital research publications library incorporating a number of automatic citation tools to cross reference papers in the system (Lawrence et al., 1999).

For pattern matching algorithms to be more successful within the AIMS project the most obvious solution would have been to train users to be accurate in their use of a numbering schema or styling when authoring. This was an unrealistic idea to try in the circumstances. The pressures of work would mean it would not be a high priority to users. Placing links using style markup might be more feasible in some industrial settings where there may be a more frequent use of templates and standard documents but not with AIMS. In the context of the AIMS system it was not worth trying to create any more patterns for links and the experiment was not continued.

One crucial idea did arise from this work. The notion that a link service would need to somehow change the way it processed a document in order to place different types of links depending on particular criteria. It was finally implemented much later as the context-based link service described in Chapter 5.

3.3.3 Compilation of Links Versus Dynamic Linking

The emphasis with projects relating to Microcosm was that documents were dynamically linked. In the Microcosm system a document viewer would request links from the rest of the system for the opening document. This philosophy was correct for Microcosm and became an assumption in implementations of the DLS. This resulted in obvious scaleability and performance limitations in the proxy implementation of a link service. For the AIMS project the requirement for dynamic link creation was not so easily justified. The major reason was a desire for the user not to need a Web proxy to view AIMS and to give the effect that the links were seamlessly part of the system. Another major factor is that the documents in AIMS would not change very often. The linkbases in use with AIMS would not change very often either therefore it was not strictly necessary to dynamically add normal generic links to the documents.

An experimental system was implemented to attempt to pre-compile links into AIMS. The goal being to batch process the linking of content, possibly at the document import stage. It was envisaged that a daemon process could maintain links and update them as required. A number of technology barriers made implementation costly. Without resorting to the low level C++ API for Lotus Notes it was not possible to access the raw rich text content of the body of a document. Therefore it could not be given to the link service for linking. Another method had to be found.

An experimental system was written to act as a Web client and 'pull' the body of AIMS documents through the link service Web proxy. The linked HTML content could then be stored back into the Notes database and displayed instead. The implementation is described in detail below.

1 A new Lotus Notes Form and View were created for the system. These showed normal documents to be displayed with the navigation parts of the document stripped out. Only the HTML form of the original document is shown. This was first designed as a way to allow PDA's and text based browsers such as Lynx to view the system.

2 A Java application was written to act as a simple Web client and make the appropriate HTTP requests to the server via the link service proxy. The application was run within the Lotus Domino server so could access the underlying database. It could obtain a list of unlinked documents from the database and hence know which documents to retrieve and link.

3 The application requested the bare version of each document by talking HTTP to the server. The application used the Webcosm as a proxy in the normal manner and as each document was requested the link service would automatically add generic links. The application would now have the raw HTML for the body of the document complete with generic links.

4 The HTML was stripped of header and footer information and placed into a new field in the document and stored in the database. The system was configured to show this data rather than the original body of the document. The server could be configured to not convert this field data to HTML but just pass it straight through to a browser.

The final result was that a user would view documents enhanced with generic links but without needing to use a Web proxy.

Whilst being an educational system building experience, the result was not a viable solution to put into production. Unfortunately it was not possible to implement the system in any kind of reliable way. The largest obstacle was that it caused numerous internal database problems and the design would not scale well. The experiment was not continued as the focus moved to attempting a more direct way to make use of the link service.

3.3.4 Incorporating the Link Service within AIMS

The experiments in integrating a link service with the AIMS system would always be compromised if the work relied on the proxy implementation of the link service. The attempts to compile links directly into the AIMS documents using the proxy had been shown to be feasible at the cost of a highly complicated and unsatisfactory implementation. Therefore ways needed to be found to work at a lower level with the technologies to ascertain what level of integration could be achieved. The obvious place to implement the link service so that it was invisible to users was within the Web server itself. This was attempted next.

Webcosm was implemented as a Web proxy component and a link server engine. These components answered requests on ports allowing for the product to be distributed across a number of machines. The link service supported a simple protocol for querying its linkbases. It would also take text, parse it for generic links and return the links. Therefore an opportunity existed to write programs that would communicate directly with the link server process.

The Domino system allows for a developer to add programming to system events such as the point when a document is requested by a browser. Each time a document is requested from the server an application has the chance to run and rewrite any data in the document. Once this has run the temporary version of the document is sent to the HTTP server component for conversion to HTML and out to the browser. The plan was to utilise the link service at this point in the request process.

An application was written that, at request time, obtained the text of the body of the document and passed it in a request to the linkserver engine. The linkserver parsed the document for generic links and returned the HTML it would have placed into the document to form a normal URL link anchor. Figure 3.7 shows an example of this.

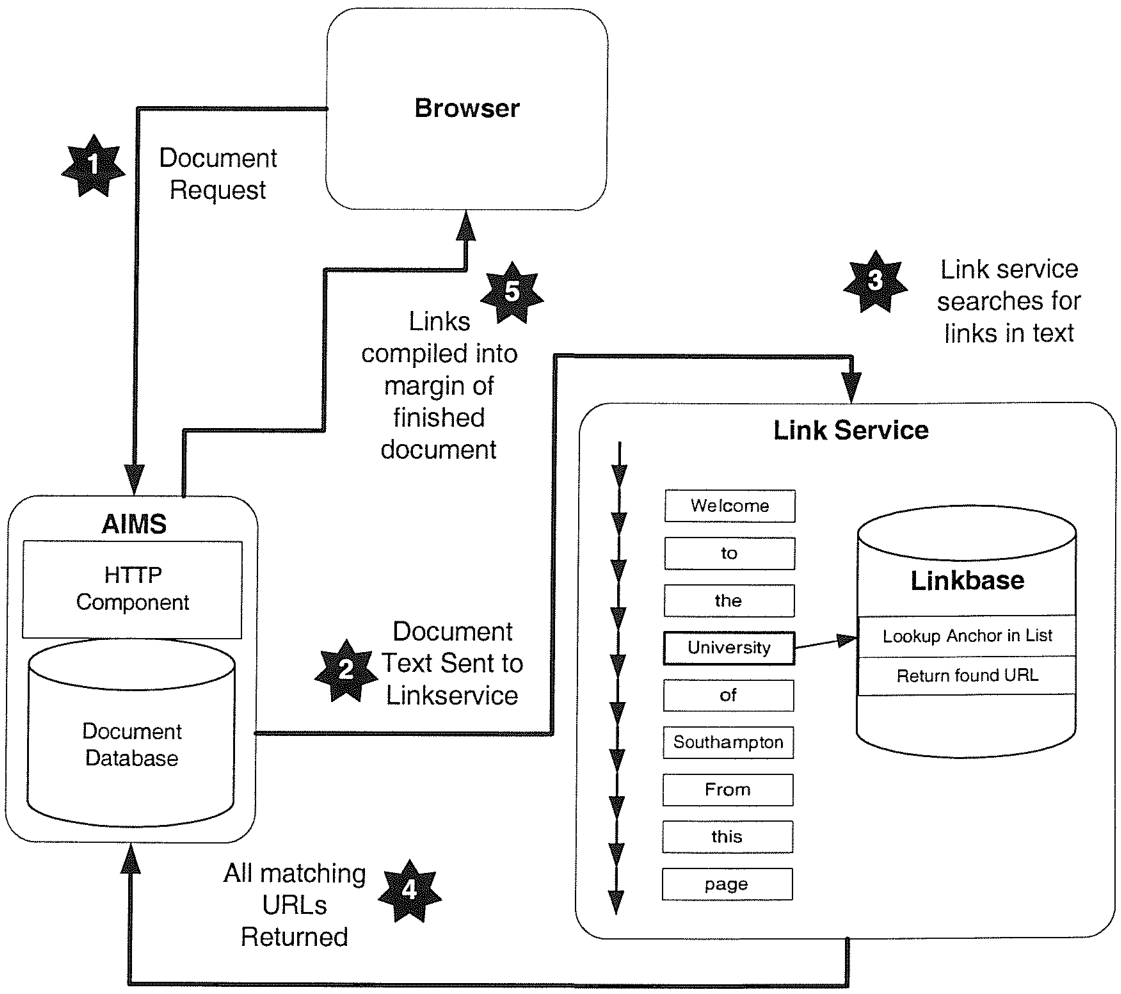
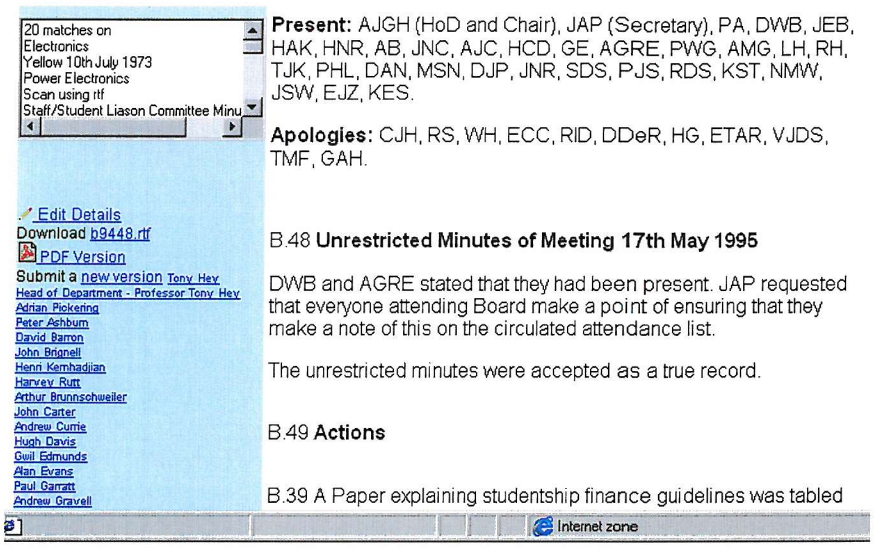


Figure 3.7: The Webcosm DLS Being Utilised as an Offline Link Service.

The Java-based application collected and processed these fragments of HTML. The application sorted and removed duplicates before parsing the HTML to build link objects. These were then used to write out a new block of HTML code listing the links. The application embedded this into a field in the Notes document and hence the Web page. Once link processing had completed the document was delivered to the browser by the Domino server. The links appeared in the margin or at the bottom of the page as a list of recommended sites for the user to view.

Figure 3.8 demonstrates one way of implementing the system. The links on names of people mentioned in the minutes have been added to the bottom of the left hand margin.



20 matches on
Electronics
Yellow 10th July 1973
Power Electronics
Scan using rtf
Staff/Student Liason Committee Minu

Edit Details
Download b9448.rtf
PDF Version
Submit a new version Tony Hey
Head of Department - Professor Tony Hey
Adrian Pickering
Peter Ashburn
David Barron
John Brignell
Henri Kemhadjian
Harvey Rutt
Arthur Brunnschweiler
John Carter
Andrew Currie
Hugh Davis
Gwil Edmunds
Alan Evans
Paul Garratt
Andrew Gravell

Present: AJGH (HoD and Chair), JAP (Secretary), PA, DWB, JEB, HAK, HNR, AB, JNC, AJC, HCD, GE, AGRE, PWG, AMG, LH, RH, TJK, PHL, DAN, MSN, DJP, JNR, SDS, PJS, RDS, KST, NMW, JSW, EJZ, KES.

Apologies: CJH, RS, WH, ECC, RID, DDeR, HG, ETAR, VJDS, TMF, GAH.

B.48 **Unrestricted Minutes of Meeting 17th May 1995**

DWB and AGRE stated that they had been present. JAP requested that everyone attending Board make a point of ensuring that they make a note of this on the circulated attendance list.

The unrestricted minutes were accepted as a true record.

B.49 **Actions**

B.39 A Paper explaining studentship finance guidelines was tabled

Figure 3.8: Integrating AIMS and the Webcosm DLS to Provide Margin Links for Documents.

This implementation is a vast improvement over previous integration attempts but is still a compromise. Although the links are not embedded in the main body of the document the system dynamically incorporates links on demand. This is approximately half way to the ideal solution of seamlessly including generic links in the body of documents. The design allows for some flexibility, the HTML generated can be varied and its position on the page can be altered so that the links or annotations can be best placed to suit the document design.

There are problems with the implementation. Again there is the usual balance between the advantages and disadvantages that dynamic linking brings. In this case the situation is made a little worse by the considerable amount of processing required to extract the content of each document and send it to the link service then embed the data back in the document for delivery to the browser. There is also a limit to how much data can be sent in one transaction with the link service and many of the documents in AIMS are very large and would break the limit. The large documents also lead to large numbers of links, albeit many may be duplicates, which can lead to the display being corrupted if dozens or hundreds of links are placed in the margin. The obvious next step in the experiment would have been to apply the compilation style techniques described earlier. The documents and links do not change often so links could be generated once and kept in the documents. Each night the system could compile links into new documents or refresh them all if the linkbases change.

Overall it was not a usable system because too many documents in AIMS were too large spoiling the effect. However a large amount of technical progress was made in link service integration. A wrapper was formed around the raw link service making its facilities available through a Java API. This alone made the effort worthwhile as it produced a useful code base used in projects outside the scope of this thesis. The result was code that gave the ability to separate the delivery and display of links from the system that it was being used with. An important step if the work was to be applied to systems other than AIMS.

In such an implementation the limitations of generic links soon become apparent. Too many links overwhelm the reader by filling the margin of the page and are a distraction. This problem has been addressed by work done for the Open Journals project by modifying the DLS. The DLS has a concept of link priority schemes, different links can be given a priority by their author and the system can colour the links accordingly or even not display certain priority levels of links.

3.3.5 Altering Links by Date: an Initial Experiment with Context

If a link service was going to add value to the AIMS system then it needed to do more than just deliver glossary-style links with no thought behind why they were there. It is so easy for the effect to be ruined if the links included with documents are not included with care and explanation. At this point it was clear that the opportunity existed to provide a system that took a step further into integrating the document database with the link service and provide links that had some contextual reason for being there. To this end a first experiment was made to filter the links provided by Webcosm according to a crude form of context.

Because the AIMS system acted as an archive for the Department an obvious contextual factor to use was to vary link delivery as the date of the document varies. There were many time-based events that could be used with the system and could be represented in the links. These included people leaving the Department or people changing roles.

A simple example is the role of Head of Department which was represented in documents as the acronym 'HoD'. The majority of the minutes stored in AIMS were written whilst the Head of Department was Professor Tony Hey. Therefore any occurrence of the acronym 'HoD' in documents should be linked to the home page of Professor Hey. But roles have changed and the link service needs to take account of this. Ideally the link service should understand the date of the minutes being read by the user and add a link on occurrences of 'HoD' to the person who was Head of Department at that time. The implication of this is that the link service understands the document enough to understand the implication of its date.

In order to achieve this effect the links need to be designed to have a time frame and the link service needs to understand how to display the correct links. It would need to be able to ascertain the date of the document the user is reading and then decide which links to display. The definition of the link from 'HoD' to 'Professor Hey' needs to include the start and end dates of his tenure. It follows that there needs to be multiple occurrences of the HoD link in the linkbase to cater for the new HoD and previous occupants of the title. If a more complex link model was being used then this would not necessarily be the case but the idea was to use the Webcosm system unchanged and merely filter its output according to a simple algorithm.

An illustration of the need for such a system is that if the linkbase of user names is generated from the live accounts system then anyone who has left the Department will not be linked. This can be solved by using an archive of personnel records to find the starting and leaving dates for employees. However there is a problem with such a plan. The information needed to make the correctly dated links, such as a personnel archive, needs to be found and accessed. The information might not exist and if it does then access might not be granted. After all it must be remembered that the Department has no electronic archive so it would be a mistake to assume that the data exists.

A first attempt to solve this problem and explore the resulting issues was created. The simplest way to add contextual filtering by date into the existing application was to add another two properties to each link - a start date and an end date. These properties define a date range or time frame over which the link is valid. Links are allowed to be open ended in one direction of time. This allows for links to current definitions in which it is not known when the definition will change. The service will return all valid links within the time frame. This allows for overlaps in time frames.

There are a number of scenarios for using links with a date range. One is to use the date of the document as the input to a link filtering algorithm. Each document in AIMS has a date as part of its metadata, this is the date of when the document was originally created or when the meeting occurred. Such a system would examine each link and only allow links to be added where the date of the document is inside the date range of the link. The system displaying the links could obtain this information from the server and display the links that were current for that time. For instance links about people who were not members of the Department before the time of the document would not be delivered to the user.

A more unusual scenario is for the user to tell the system what date they wish to use. The user 'turns back the clock' to a specific point in time and browses the system viewing the state of play of the Department at that time. This could be extended to the AIMS application so that the server only lists the documents that had been created by that time. This version was never implemented but was found to be technically possible. Such an application could have uses for certain archival database applications.

To test how possible it would be to create a basic link filtering system modifications were added to the system that communicated directly with the commercial Webcosm link service. The existing system was designed to parse the HTML fragments that represented each link that the link service returned. Without having the ability to modify the link service the only way to add extra parameters to the links was to create special description strings with a certain format of description followed by 2 dates. The system parsing this HTML could extract the dates into new date fields of the link objects. From there it was a simple matter to provide a 'barebones' contextual link service.

This application demonstrated that the idea of a link service with the ability to provide contextual linking was worth pursuing properly. The actual implementation was not worth expanding on as full control was needed of the link service or the project scope would be severely limited. Therefore it was necessary to find or build another link service to use for further work. Before that could begin a considerable amount of thought was needed to understand and design a system that could be called 'context aware'. The next chapter describes this work.

3.3.6 Performance Issues for DLS Applications

The major drawback of the proxy implementation approach is in performance, especially with regards to scaling. The server must process and add links to the whole of each document that the user views using their browser. The performance is governed by the number of users using the service and the number of links available.

The other major usability issue with Web proxies is that a Web browser application must be configured to use the proxy. This has always caused problems in applications of the DLS and its descendants. (Hitchcock et al., 1998) summarises the problems well when reporting on the conclusions of the Open Journals project, the first major usage of the DLS.

> Technology must be transparent: users want better services without having to install new software or change computer settings.
>
> It was common practice for Web users, especially back in 1995 when the project began, to download and install software from the Web to improve the Web experience. With the exception of Acrobat it seems the practice of software download does not apply to typical e-journal users, as the project soon discovered from publishers and librarians. So the link service software was rewritten to work at the server end, mediated by a user-set browser proxy. Even this was insufficient. Libraries do not want settings on shared machines to be altered, and proxy settings can interfere with firewalls in corporate environments. In the latest version the direction to the proxy server is attached to the URL, leaving the user to browse the Web conventionally and do nothing to receive the link service, bar starting from the right place!

Hitchcock is referring at the end to an implementation of the DLS as a form of portal. The service is used as a starting page and any pages navigated from that page will have all link destination URLs augmented by the service in the following manner.

If a link was to *http://real-server/real_page.html*

then it would become *http://linkservice/augment.cgi/http://real-server/real_page.html*

This does not require a user to set up a proxy but does require a user to start from a home page hosted by the link service to make use of the system. In effect the link service acted as a form of portal, a concept not yet in use on the Web at the time. This was an acceptable solution for their situation.

There are a number of answers to the perceived problem of implementing a link service as a Web proxy. The simplest is that this is scientific research and not a polished product from a major software company. Some 'rough edges' are to be expected that should not detract from the aims of the research. This argument has become much weaker when the DLS or its derivatives have been commercialised, as Webcosm, or implemented outside of the laboratory.

The implementable answer is to make the functionality available at either the server end or the client end. The server end requires that the link service and the Web server be integrated in order to be perceived to be a single system to a user. This was attempted with Webcosm which was implemented as a plug-in to the Apache Web server. It has also been attempted with the AIMS application, see Section 3.3.4.

There are two ways to implement a link service for the Web at the client side. As seen previously the first is to modify or add hypertext functionality to the Web browser. This was how the DLS began life with an interface addition to Netscape. This proved unsustainable as the APIs were unstable during the early versions of the browser. Modern Web browsers are highly extensible and many similar systems have been created in more recent times.

The second way to implement a client-side Web link service is to write a personal proxy. This has proved a popular implementation strategy for a wide variety of applications, not just link services. Whilst the user needs to go through the process of configuring the browser it can be argued that if they are motivated enough to use the software in question they will not be troubled by this task.

The WBI system, described in Section 2.3.3, included a personal proxy implementation. The Muffin (see Section 2.3.4) modular Web proxy used in the implementation of my work was also originally designed as a personal proxy. The original use was a system to purge annoying artifacts from Web pages such as adverts and Java applets.

The difference in the implementations is an important factor. The DLS as a proxy is more effective for users of the Web because it does not require physical integration with the browser and is platform and browser independent. Documents are just altered for them and require no user effort to activate the system once the proxy is in use. The first implementation required the user to actively ask the system to return extra links by using a menu item. Whilst this type of behaviour was normal for Microcosm and other hypermedia systems it was not compatible with the simplistic hypertext navigation facilities of the Web where a user simply clicks on blue underlined words to follows links.

As the history of the DLS shows there was a time when the only reliable way to augment a Web page was to use a proxy. Operating systems and Web browsers have improved to the point where this is not the case, as long as a developer is not concerned with providing a cross platform solution. My decision to continue to use the proxy as the main implementation framework is purely for convenience, the systems and principles described could be implemented again using other approaches. The most obvious would be to write a new toolbar for Internet Explorer.

3.3.7 Summary

The experimental integration of AIMS with either the DLS or Webcosm led to a number of conclusions and requirements. The chief of these was that a new design of link service architecture was needed in order to allow for new methods of creating tailored links. Compiling links into documents was a seemingly sensible option for the static documents of AIMS but this implied the use of standard links as opposed to the more creative linking approaches that could be produced if the link service was a dynamic system. The current system designs hampered more creative approaches such as pattern matching text to find links or attempting to compute links by using the date of the document. In both cases there was no clean way to alter the underlying model of the link services. The services provided one model of links and one way to compute the link to place on an anchor.

Providing link services within the document management system showed promise as it did not require a client to use a proxy but again highlighted the need to be more creative in linking. The service was providing too many links and links that were of little use to readers. The unsatisfactory implementation methods also restricted the experiments.

Linking on dates was a first proper contextual linking system. It demonstrated that a properly implemented link service could use new ways to compute the links that should be placed in documents. Linking on dates also showed that a new link model was required in which links had date ranges over which they were relevant. The experiment demonstrates a requirement for a close union between a link model, the situation of use and the code that uses these to compute the final link. Describing the situation of use, or the context, is the subject of the next chapter.

3.4 Evaluation of the AIMS System

3.4.1 Formal AIMS Evaluation

A formal evaluation was carried out to measure the acceptance of the AIMS system by the secretaries of the ECS Department. The evaluation focused on the subjective opinion of the people and measured the time taken to perform the major functions of the system. This work was carried out for the AIMS project by Dr Gary Wills (Wills et al., 1999). The report presents the methodology and describes the rationale behind the approach used. A summary of the report and its results are included below. In this section references to 'users' means staff who are 'contributors' to the AIMS system and not just 'readers' of the Web pages.

The evaluation was conducted in two stages, first a contextual review (Preece et al., 1994) of the working practices of the secretarial staff was undertaken. The second stage, was a usability study of the system, which was conducted in two parts. A structured expert review using discounted usability engineering (Nielsen, 1989, Nielsen, 1994) with an additional principle of 'Provide Navigational Aids' to ensure that the hypertext components of the system are reviewed (Wills et al., 1997). The second half of the usability study was to conduct a time trial with users completing questionnaires. The questionnaires were used to measure the user subjective opinion of the system and their acceptance to use the system (Davis, 1993, Davis & Venkatesh, 1996), an additional criteria of Navigation was added to measure the users opinion on how well they could move around and through the information space.

The contextual review showed that currently the main method of archiving is in the paper format, making it difficult for academic staff to obtain back copies of meetings to check on discussion or discussions made at previous meetings. The length of time required to photocopy, collate minutes, put them into envelopes, stick the address labels on (which have previously been printed) then walk them to each group, takes on average between six to eight hours.

Task	**Average time second (SD)**	**Median**
Task 1 Enter a new document	98.6 (10.0)	98.5
Task 2 Enter the next set of Minutes	60.8 (7.7)	63.5
Task 3 Edit document details	33.0 (6.3)	35.0
Task 4 Search for a document	40.9 (12.0)	43.5
Task 5 Using Webcosm to follow links.	31.5 (8.3)	32.0

Table 3.1: Average Times to Complete the Tasks

The average time to complete the tasks is shown in Table 3.1. The time taken to complete the tasks was consistent throughout the user group. The obvious saving of time comes from the ability to publish the set of minutes extremely quickly, when compared to the current method. The ability to disseminate information quickly would be a significant cost benefit to the department. However, this will only be an advantage if there is a paradigm shift in that an academic member of staff will need to read the minutes online, or as last resorts print them off themselves to read them.

The general comments from the users showed that they were interested in the idea of a central repository for administrative information. This was reflected in the high score from the questionnaires given to the Impression Category in Table 3.2, and the 'intention to use' category from the Technology Acceptance Model criteria in Table 3.3. All the users replied that they would recommend the system to their colleagues, and that the majority (87%) could navigate the information space.

The scores were normalised by dividing the score by the number of respondents and the number of questions they answered. The maximum normalised score is +1 indicating a very strongly agreed to -1 indicating a very strongly disagreed to all positively phrased questions (and visa-versa on all the negatively phrased questions). A score of 0.5 indicates that all the users agree with a positive statement, i.e. They agreed with 'The AIMS system is one that I want to use on a regular basis.' The questionnaire also showed that the scores given were not affected by the user's access to the Internet from home, or the user's preference for using the Internet.

Criteria	**Normalised Score Range -1 to +1**
Impression- *user's feelings or emotions when using the software.*	0.50
Command - *the measure to which the user feels that they are in control.*	0.46
Learnability - *the degree to which the user feels that the application is easy to become familiar with.*	0.44
Navigability - *the degree to which the user can move around the application.*	0.43
Helpfulness - *the degree to which the application assists the user to resolve a situation.*	0.43
Effectiveness - *the degree to which the user feels that they can complete the task while using the system.*	0.37
Over All	0.44

Table 3.2: Score from the Questionnaire

TAM Criteria	**Score**	**Normalised Score Range -1 to +1**
Intention to use	17	0.57
Perceived Usefulness	41	0.46
Perceived Ease of Use	80	0.41

Table 3.3: Technology Acceptance Model Scores

3.4.2 Reflecting on AIMS

At the time of writing the AIMS system had been available in the Department for over 3 years. It is still in use by the central administration group of the University and archives the minutes of their meetings. They have been using it for 3 years and are currently looking for a replacement solution within a wider remit for a University wide document management solution. It was also evaluated by a number of other UK University departments.

Within my department the formal usability evaluation was highly encouraging but the system was never widely used. More importantly for the author it was extremely successful as a research project and testbed for this PhD work. The Lotus Notes development skills have been put to great use for a variety of small, Web based, applications for other projects and administrative uses. However the level of skill needed to produce them was only gained after many years of struggle against an extremely complicated system to learn. This complexity is one of the factors which has contributed to the lack of official interest in taking over and maintaining the system. It is fair to say that Lotus Notes has gained a little notoriety for its 'unique' interface design.

The design of AIMS had one serious technical flaw. It struggled to cope with very large documents or documents that spanned multiple files. Despite many attempts no satisfactory way was found to present a decent user interface that allowed users to relate documents together to indicate that they were related. For example a way to show how chapters of a book were related. A great deal of work went into ensuring that the underlying system could cope with 'compound' documents and even to cope with complex version control scenarios where users could update certain, but not all, chapters. This was never fully deployed. The AIMS application pushed the capabilities of Lotus Notes and early Web browsers to the limits and often it was not possible to achieve the effects required. Many of the early link service integration experiments were hampered by the inability of Notes to do what was needed.

What has changed in the time since AIMS was started is that a lot more administration information is being published on the Department Web site. This is being done by the Web Masters acting on the instructions of senior members of the Department. A variety of other sources have also appeared including a publications database and a central database of research projects is in production. In the vast majority of cases the information is there partly due to the much greater usage of the Web and also because external factors such as Teaching Quality Assessment provide a real need. What has not happened is a model where the producers of documents are the Web publishers of their documents. This unusual model is allowed by AIMS but the hierarchical nature of the Department expects that a mechanistic task such as this should be done by purposely employed administrative staff. Many secretaries were trained on using AIMS but few ever used the system on more than a few occasions despite their positive response at the time. The vast majority of information in AIMS was placed there by myself.

In the time since the AIMS project began some of the working practices of the Department have not changed. The minutes of meetings are still word processed and printed then distributed by hand. They are still not available online and there is no sign of this changing. Therefore the author feels justified in the assumptions made that the system would have to be incredibly simple to use or it would not be adopted. Even though submitting a document takes very little effort the system is not used. The combination of little change in some aspects of the Department, little demand for the material to be online and the employment of more Webmasters to specifically author Web page versions of the certain documents has meant that the AIMS site is little used. This is regrettable but the focus of this research work lies in the link service provision ideas that were provoked by working with AIMS and the real world documents it contained.

3.4.3 Managing the Service

As the system design evolved and the system was brought into service it was found that there were two distinct roles for managing the system. The first was technical. The system administrator needed to have a certain amount of Lotus Notes knowledge which is a specialised field. Many tools and scripts are provided with AIMS to maintain the documents and the database but using them effectively takes understanding of the database model and the programming languages involved. Lotus Notes features a language called LotusScript which is very similar to Visual Basic and later provided support for Java. These languages provided access to the large object oriented API of the system which requires a considerable amount of training to fully grasp. The server administration tools shipped with Lotus Domino are powerful and well documented but it is not a trivial task to understand the full complexity of the system and fix problems.

The second, vital, role is editorial. This role is crucial in keeping the site up to date and relevant. As well as expecting all secretaries and administrators to actively submit their work to the server a person or persons is needed to ensure that other documents are in the system. There is also a considerable amount of work to do because not all staff will actively use the system. That person needs to know what documents exist and where to obtain them electronically to ensure they are in the system. It may also be that the submission is to be a complicated collection of documents rather than one single file. For example the major, regular, submission of documents follows the meetings of the Departmental Board. The minutes, agenda and accompanying reports are submitted to AIMS together. There are many authors and the system works most efficiently if one person ensures that all documents are submitted. The logical person to do that is the meeting secretary.

This need for a second person to manage the service, an editor, contradicts the initial goal of the project. It was hoped that each person in the Department who wrote something worth saving would directly contribute it to the system themselves. The system is simple enough for that to happen. Unfortunately it never did. One reason is that the service was slightly ahead of its time with the users not being as Web-aware as it needed. Another is that academics will not do 'menial' tasks if they do not have to and will expect an administrator to do it for them. The third is that administrators will only do precisely what they are told to do for a job and as the political will has not being there to instruct them to do it they have not entered documents. Since the formal evaluation was carried out no further training has taken place and no new administrative staff have been trained. In short, using AIMS has not entered the culture of the Department.

3.5 Conclusion

This chapter has described the AIMS document management system. This was developed to allow writers of normal word processed documents to submit them directly to a Web site which would take care of all the work of publishing the document on the Web. This design meant that none of the pages would have links within their content so the application of technologies such as the Distributed Link Service and the commercial equivalents were highly desirable. From this base point a variety of prototypes were built to evaluate how best to integrate open link services with a Web server and try to improve the quality of linking available by closer integration of the two systems. This chapter has described the ways in which the two systems were integrated to alter link delivery depending on factors such as the date of the document in question or by pattern matching.

Near the time that the AIMS project formally ended the conclusion had also been reached that a link service needed to be built that understood how to provide links in context. The second conclusion was reached that in order to build such a system the service would need to be designed for contextual link processing from the beginning and not as an afterthought. At this point the use of the DLS and Webcosm stopped and the planning began to write a new contextual link service.

Before describing the resultant contextual link service architecture in Chapter 5 the next chapter tackles the difficult topic of context.

4 Context

4.1 Introduction

The experiments with integrating the DLS and its commercial descendant, Webcosm, now Portal Maximiser, with the AIMS document management system resulted in the decision to move the design of such systems in a new direction. The link service needed to deliver links with greater relevancy to a reader whilst not overwhelming them with too many links or distracting them with irrelevant destinations. In summary a link service should provide contextual linking and be adaptable to new uses and types of Web sites.

Some of the integration approaches taken with the AIMS system resulted in the user being presented with dozens of links to people or other information, sometimes delivered in a small space such as the left hand margin of a Web page. Clearly the links needed to be reduced in number. The natural approach to this problem is to filter the links that are delivered with a document. They would need to be filtered according to criteria relevant to the circumstances. This is where the work on improving the link service overlaps with the field of information filtering (Belkin & Croft, 1992, Foltz & Dumais, 1992, Loeb & Terry, 1992). Deciding what criteria to use when filtering is where this work strays into the much broader field of context.

Filtering links is the simplest solution, and the most crude, it assumes that the system has no influence over what links are being created and placed in the system, merely that it blocks delivery of a link at the last moment. Microcosm showed that a link service infrastructure can do so much more. The next logical advance is to generalise the link creation process itself. This is the process of parsing documents and making the decision of what links to place where, what form they should take and how the user should interact with them. The system now becomes a contextually aware adaptive hypermedia system. Generalising further we see that the function of the link service could be to provide content, usually links or data to be used as links, to place into Web documents. The system must be designed in such a way that it is as general as possible for the purpose of serving links. This chapter discusses some of the ramifications of this goal.

The difficulty in attempting to improve the design of the system is that one must confront the problem of generality identified by John McCarthy (McCarthy, 1987). He argued that even a small addition to the predetermined possibilities that a program must handle often results in a rewrite, starting with the underlying data structures. Any system that attempts to utilise context requires a model and it is this model that can always be broken. Bruce Edmonds points out (Edmonds, 1999) that in order to pragmatically get on with building a system based on a complex model many factors must be ignored and a set of assumptions made. Again this results in the possibility that new circumstances will be found to break the model. Edmonds goes on to illustrate methods for modelling context to use in applications. The key skill is to learn what factors are so constant they can be ignored and to understand when other factors will affect the model.

Ideally a generalised link service design would be able to cope with new situations, new document types, new link types and new methods of processing links. It should be possible to change the underlying model used by the link service to use new input factors and situational contexts. The ideal version of such a system is impossible but in order to make some advance in the right direction we need to understand how systems claiming to incorporate context do so. This requires an investigation into what context means to different disciplines and how researchers pragmatically attempt to take advantage of it.

4.2 What is Context?

Context is a word meaning many things and is routinely used in connection with a vast array of subject areas. In computing the term has dozens of uses and we shall look at a number of the major areas that describe themselves or their systems as contextual.

It is so difficult to precisely declare what context means that it is traditional (Akman & Surav, 1996) to start with dictionary definitions:

> "The weaving together of words and sentences."

Oxford English Dictionary.

> "The circumstances in which an event occurs; a setting."

The American Heritage Dictionary of the English Language, Fourth Edition.

> "The interrelated conditions in which something exists or occurs."

Merriam-Webster Collegiate Dictionary.

Another set of definitions and usages of the word can be found by using Google. Searching for "What is context" gives an insight into how the only thing that people can agree on is that nobody can agree on a definition. This is left as an exercise for the reader as the list will change over time.

This problem is acutely felt in computer science. Bruce Edmonds puts it well (Edmonds, 1999).

> "Frequently at workshops and conferences on context, one finds that the emphasis is on drawing distinctions between different types of context and illustrating how little each type has to do with the others."

In her keynote at Information Seeking in Context (ISIC96) Brenda Dervin said that to

> "ask [What is context?] turns out to be almost embarrassing, and certainly a question leading to a quest that demands extraordinary tolerance of chaos." (Dervin, 1997).

She notes the overuse of the word in the social sciences, a phenomenon clearly spreading into computing.

Dervin's survey of context (Dervin, 1997) forms a continuum in the treatment of context. The extremes of the study of context can be caricatured in the following ways.

Everything is a context which must be measured and taken into account. This reduces context to a set of attributes to be measured and it is possible to find approaches where 'almost every imaginable attribute of people, culture, situation, behaviour, organization, or structure has been defined as context'. Context is the enemy of understanding, to be eliminated or reduced to simple data.

Context is a continuum. The view states that using context is the only way to understand human behaviour. It is the way in which meaning is transported. It is not possible to quantify this amorphous whole as each context is unlike any other, derived from an unlimited set of factors.

These extremes of opinion essentially reduce to taking either a quantitative or qualitative approach to defining context. Dervin's survey picks out many major themes in all of the areas studying context, some of which are instructive in this discussion.

One of the themes is that knowledge is partial and temporary. Therefore any attempt to use context should cope with uncertainty and fluidity in the model. In some ways a context is a historical device, a snapshot of a system in time. A core theme is that the knower and the known are bound together when producing an understanding. This goes along with the idea that context can never usefully be treated as an independent entity and that doing so has the effect of reducing context to something more simplistic.

Ultimately there is a balance to strike. Some method must be used to measure, define or determine context at the risk of reducing an essentially unmeasurable whole to something overly simplistic. In order to pursue practical goals where context is concerned one has to 'make a cut' somewhere, somehow. This is especially true in the highly pragmatic circumstances of the work presented here. Happily there are many practical lessons to learn from diverse fields on how to go about it. We shall now see what computer science has done in order to represent, determine and utilise context.

4.3 Representing Context

To use context in some form in an actual computer system it must be modelled and represented. Here, again, we find a bewildering array of approaches which have a lot in common with the need to define context and use it. It is most normal to find models that are solely to satisfy a particular need or application area. Applicability to other fields is often not possible nor envisaged. This is born of the necessity to implement something that is not too general (Edmonds, 1999).

More detailed definitions of context in computer science fields require more specialised fields. The definition of context must be given in the context of the subject being discussed. Perhaps this is the root of the problem of overusage of the term.

In the AI field (Akman & Surav, 1996) give a comprehensive review of various attempts to formalise context and identify what role context plays in a variety of fields in the subject. These include intelligent information retrieval, knowledge representation and natural language processing. In natural language processing the need is to narrow down the possibilities of meaning in something such as a statement or sentence. The aim is to remove ambiguities from language and attempt to eliminate multiple meanings or at least try and attach some form of probabilities to each possible meaning. This may include attempting to find the common ground between speakers, the intrinsic context of the conversation. Some utterances must be accompanied by context - "the library is to the left of the administration building" is only correct for a single location on a university campus and that context must be supplied for the sentence to be useful.

The need to disambiguate is also true of categorisation studies and methods of representing knowledge and reason. The process of categorising the everyday world requires context which is often implicit in our language but needs to be defined if a system is to be even remotely effective. A temperature may be classified as 'high' for the air temperature but 'normal' for body temperature. This requires rules to be written, which, as stated earlier, can always be broken if one tries hard enough.

McCarthy's introduction of an idea for formalising context (McCarthy, 1987) was based on the assertion that there can be no general context where everything is meaningful and all stated things are true. Therefore the domain or context in which something can be stated must be defined. The basis for his representation proposal was that a proposition *holds* in a certain *context.*

The various methods that (Akman & Surav, 1996) cover mostly concentrate on the notion of narrowing the domain of possibilities to make it more possible to model context. The overall goal is to create a database of common-sense knowledge that can be re-used outside of a specific domain. For instance defining micro-domains in which a context is defined and can be used. The concept is crudely analogous with local variables and functions in programming languages. The different approaches do not always agree, for instance one says that an assertion always holds, in every context, but that the values might be different.

In order to be able to learn from a situation, or make sense of it in a model, Edmonds advocates that we need to make assumptions that many possible causes of an event remain constant. Therefore they can be factored out of a model of a context as parameters. The list of assumptions made varies with the domain. For instance these could be physical factors such as coordinates in a physical model - time, distance, gravity. At the other end of the scale in terms of complexity is the world around us and our social interactions. Here there are so many factors that it might not be possible to find any commonalities to use.

A highly practical example of this is described by (Turner, 1997, Turner, 1998). He describes the development of operating software for a robot submarine which is designed to operate autonomously. To do this the team developed the agent based software to model its world as a set of contexts. In order to be able to produce these the factors and inputs have to be limited to those that differentiate the different contexts that the vehicle can find itself in. Their terminology differentiates situation from context. Situation is all the circumstances surrounding the agent including internal system state.

Context is the set of elements of the situation that impact behaviour. Each context tells the system how to operate under the conditions found. Again, the model is reduced to just the factors that are deemed to matter to the application.

The system attempts to find all of the context-schemas that match the current inputs and then merge them into a new current context. The context includes all of the rules of how to operate whilst in that context. The contexts are implemented in frames using the Common Lisp Object System and include relationship hierarchies. The problem they face is attempting to predict all of the situations that the robot could find itself in and what it should do about it. It is not surprising to learn that the vast majority of the code is concerned with disaster recovery and how to deal with failure or danger to the vehicle. The system contains fuzzy event detection capabilities to aid in matching inputs to a context. Secondly there is a way to grade the importance of an event in order to correctly match it to the right context. Thirdly they can directly link an event to a goal through a context. For instance the event 'leak' can trigger a rise to the surface if the context is 'at sea' but the context 'in harbour' will require it to sink to the bottom to avoid damage from surface traffic.

For a system to be context aware a practical way to implement a solution is to have a user model, a system model and a task model, or some combination of the three. User and task models try to understand what the user wants to achieve when using the software or system. For instance the user wants to enter a room or find relevant documents to the item of interest. As before the models must attempt to encompass a wide enough set of expected behaviours and expect to be broken at some point. (Selker & Burleson, 2000) point out that a users needs are often intangible. Users often do not behave logically and issues of comfort, self-image and motivation come into play. For instance a user might not shop online because it gives away a lot of information about their personal tastes to a third party. The user might take their grocery shopping list to the supermarket on a state of the art PDA because they love to have the latest gadgets when a pen and paper would be sufficient.

The simplest models are based on parameters and environment variables as this is often easily sufficient for the application and is easy to implement. Schilit's (Schilit, 1995) system used environment variables stored on context servers. The HP Cooltown project used a Web based context model in which an entity's context could be retrieved using a URL (Kindberg et al., 2000). In this case the system was concerned with the movement of people and things between places and the infrastructure allowed the transfer of their *Web presence* between servers representing different locations. Another location based context model is in the Sentient Computing Project from the AT&T Cambridge lab (Harter et al., 2002). The system stores people's location as part of a shared world model allowing users and software to interact together in the model. Users are tracked around the building and all of the office furniture and layout are also in the model. In these systems context is essentially location.

4.4 Determining the Context

A system utilising a model of context requires a way to detect changes in the situation which will lead to the system switching to a new context or updating the existing context model. In different subject areas the definition of context varies.

4.4.1 Document Understanding

For many researchers context is closely linked to understanding a document. One goal is to ascertain what a user is reading. For a variety of reasons many companies and research groups are working on ways to automatically determine the meaning of text documents. The major method is to mechanistically understand or categorise text. If a system can determine the subject matter of a piece of text then it can be said to understand the context of the document. Again the definition of context is different to those seen from before. It is only a slight exaggeration to say that document understanding is seen as a potential source of huge revenue to many companies. The promise of being able to accurately guide users through enormous document repositories to the documents, paragraphs or knowledge they need to know is driving a thriving industry.

One of the most common basic computational tools used is a statistical technique to calculate words that occur more regularly in documents and give an indication of their importance. Term frequency - inverse document frequency (TF-IDF) is a well established information retrieval technique (Salton & Buckley, 1988) which works on a corpus of documents. An index vector is created for each document, each element of the vector represents a term in the document and its magnitude indicates how well that term represents the content of the document. This is achieved by using the frequency of the term in the document and the inverse of the frequency of documents containing the term. Hence a term appearing often in a document but rarely overall will have a high magnitude and hence strongly indicates that the document is a good choice if a user wants to know about that subject.

The system works well for many situations but has limitations which various researchers have tried to address. The major problem is that documents need to be pre-processed and the corpus needs to be known in advance of usage. If the techniques are to be used in connection with Web browsing then it is a serious problem as the corpus cannot normally be known in advance.

A tool such as TF-IDF will aid greatly in automatically understanding the content of a document. The context of the document will require more work. Again we need to understand what researchers use as their definition of context. In this case context means how a document is being used, it is related to the reader's tasks or goals. As (Bauer & Leake, 2001) puts it 'there would be a different answer to the question "What is this document about?"'. Various methods have been developed to determine the users goal by looking at the documents the user is reading and seeing if a pattern can be found in the individual document contexts and hence the overall context of the user.

The most obvious way to obtain such information is to request a user to directly provide it. This is an unrealistic approach as users will often not take the time to supply such information so automatic methods are required. An example of such a system is WordSieve (Bauer & Leake, 2001). WordSieve works in real time to filter documents the user is reading and maintains 3 layers of information about words in the documents. The first layer is similar to TF-IDF in that it builds a list of most frequently occurring words in the documents. The system is built around the notion of the level of excitement of a term and has decay functions built in so that words that stop appearing in documents will fade from the list. The second layer maintains a ranking of terms appearing in sequences of documents. If a term keeps featuring strongly in the first layer then it will move up the ranking, the rate of movement in the ranking is also a factor. This layer does not 'forget' words as fast as the first layer. The third layer is the opposite of layer 2 in that it maintains a list of words which only rise to the top when they stop appearing in list 2. It helps to define the point at which the context has changed.

Finding related documents for a user is a popular area with similar goals to those in this thesis. Again the word context is often used in the description of these techniques. There are plenty of methods to choose from. For instance the WebTop system (Wolber et al., 2002) builds a personal space of related documents with both links to the document and from the document. They call this the context view of the document. They employ multiple techniques to provide the relationships. The system presents a personal space which integrates three sources of linking information they contend are traditionally kept separated; local file and directory information, links to Web pages in the form of bookmarks and the links found within the document themselves such as hyperlinks and citations. In addition the system attempts some document analysis of the current document being viewed or edited to ascertain its 'meaning' and passes that to the Web search engine Google (Brin & Page, 1998) for further linking.

Another example was Kenjin from the company Autonomy[1]. Kenjin was a simplified version of the full Autonomy product and ran on the user's machine displaying a toolbar in a browser, word processor or email client. It monitored the users typing and then found similar documents for the user. Autonomy claimed that Kenjin could find the real ideas within documents so that the returned documents were found on more than word similarity. The system used Baysian inference logic to interpret the context of the text in question. A user could drag a text selection onto the toolbar to directly query for results or the system could automatically supply links. Computing magazine reviews stated that searches were no more effective than a normal search engine.

The closest related project to the author's own work is the PhD work of Samhaa El-Beltagy (El-Beltagy et al., 2001). Her system provided generic links in documents where the links were placed by determining the context of the document and using generic links of similar context. In this case the definition of context was the overall subject matter of the document the user was reading. Links were created by harvesting documents and using a TF-IDF based technique to ascertain the subject matter. Hence a generic link to the document with an associated context could be stored. In order to build a database of links to use in the system, groups of users nominated interesting documents to process via a user interface agent. This allowed the system to be primed with contexts and links. The TF-IDF process alone was not enough to distinguish between certain types of words as the vectors of some types of documents would be very similar. Similarly to WordSieve a more accurate way was established to distinguish between certain document contexts.

4.4.2 Pervasive Computing

Those in the field of pervasive or ubiquitous computing use the term 'context' to define some element or elements of a person's physical situation. It is implicit in all of the work in this field that context is a physically measurable phenomenon. It can be argued that the impression given by some of the papers in this arena is that the term is theirs alone, even to the point that the community has coined the phrase 'context-aware computing'. In these systems context is a measurable dimension to be used as an input to a software application, most commonly by the use of physical sensors in the users environment. On Dervin's scale, see Section 4.2, we can see that this is a field that is near the extreme of the quantitative approach to context. Whether it is truly context is a different matter but many of these applications are the most practical applications to date and do advance the field forward more than most considered here.

[1]www.autonomy.com

Ubiquitous computing has been described by Mark Weiser (Weiser, 1991, Weiser, 1993) as the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user. Rather than the computer being a focal point at which a user must sit, computers are so embedded, small and natural that they are just used without the user realising it. The steps to achieving this require systems that understand the users physical context (Davies et al., 1998, Mynatt et al., 1997, Pascoe et al., 1998), most commonly in the form of location and the systems relationship with other resources on the network. Such systems are known as context-aware applications (Schilit et al., 1994), the subject of Bill Schilit's PhD Thesis (Schilit, 1995) and subsequent work (Mankoff & Schilit, 1997). Systems built into the fabric of offices that follow users around a building are known as follow-me applications and are seen as a subset of context-aware applications. They make use of a person's location as well as a detailed profile to automatically display a users desktop on any computer in the building (Harter et al., 2002).

A major contribution to this particular field is Anind Dey's PhD thesis (Dey, 2000) in which he describes the development of a 'context toolkit' (Dey et al., 2001). The definition of context is given as

> "any information that can be used to characterize the situation of an entity, where an entity can be a person, place or physical or computational object."

He describes 'context-aware' applications that

> "use context to provide task-relevant information and/or services to a user."

The goal of the work is to develop a framework to deal with context handling in the same way as input handling from devices. The framework is a way of abstracting the differences between context inputs to provide generic methods for developers to use in applications. Each context is abstracted primarily through the use of context widgets (Salber et al., 1999) in order to provide similar interfaces to each sensor and facilities such as reuse, polling and notification mechanism and a common interface. The design was modelled on graphical user interface widgets. This context widget idea was the starting point for the toolkit which added support for a number of more complex issues. The biggest issue when building environmentally aware applications is that sensors are often physically distributed over wide areas, such as a building. The consequence is that applications and the computers being used are also distributed. This also leads to applications where the context widget does not belong to the application but is independent. The third issue was that some contexts required greater abstraction and more processing than others before they were useful to applications.

The toolkit architecture consists of the components illustrated in Figure 4.1.

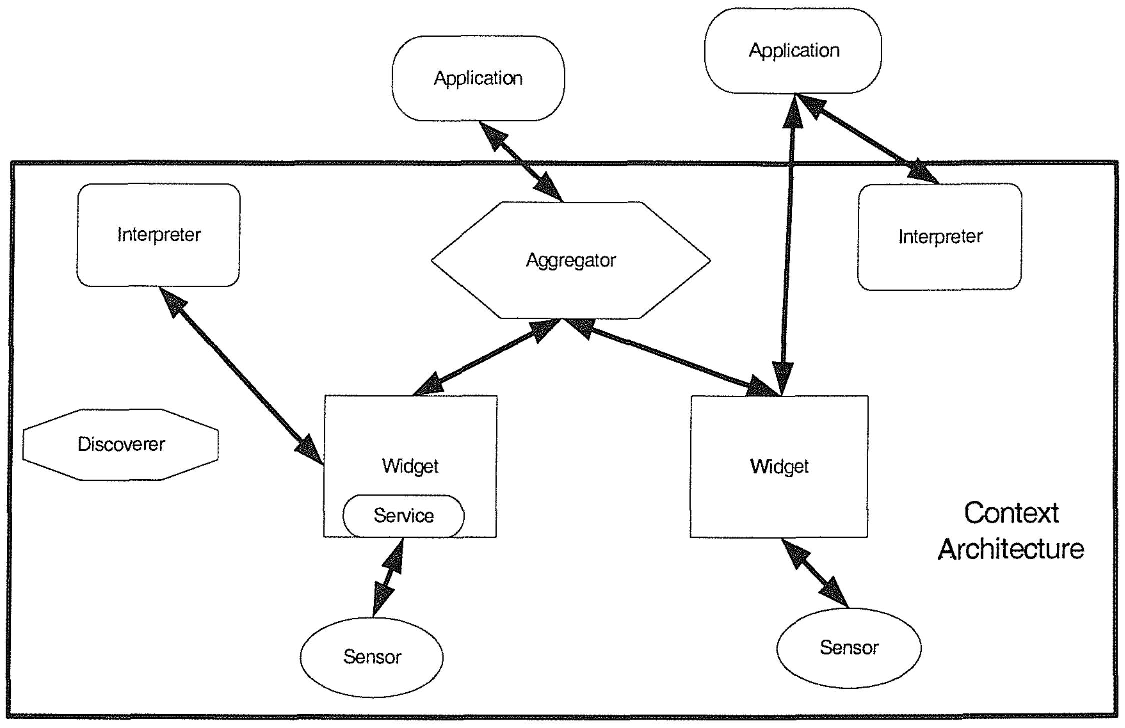


Figure 4.1: The Context Toolkit Architecture.

Sensors provide data to context widgets which make it available via attributes and standard programming techniques such as polling and callback. They also provide historical context information. Context servers aggregate context data at a higher level from multiple widgets saving a developer from subscribing to multiple individual widgets. Interpreters provide abstraction from the raw data provided by widgets, this enables code reuse and also means that applications do not have to do all of the raw interpretation of data themselves. Components register with a Discoverer on startup which act as a form of resource discovery service to the rest of the system. These components communicate using XML messages via HTTP providing cross platform support for distribution.

The overall application has similarities to a distributed agent framework such as SoFAR (Moreau et al., 2000). These provide notification and callback services with individual agents providing services to other agents in the system. The abstraction that agents and their ontologies provide are similar in capability to the context toolkit. It is also similar and could be rewritten to conform to the SOAP[1] techniques for writing distributed applications. In fact it would not be surprising at all to see future versions supporting a form of SOAP. However the toolkit is designed from the start for use with applications dealing with physical phenomenon such as people entering a room and have the correct logistical capabilities built in whereas agent frameworks tend to be designed for more general purpose work.

An application from the same arena but with a different definition of context is a context-aware office assistant (Yan & Selker, 2000) developed at MIT. The developers wanted to decrease the number of interruptions to an office owner. They noted that many visitors to an office do not want to talk with the office owner but may just want to leave a short message or check the status of the person. The office assistant acts as an intermediary which visitors can interact with on behalf of the office owner. The system combines an office door sensor system which can detect people approaching a person's office with an interactive scheduling system and the ability to interact via a display with visitor and owner. The agent understands the status of people and can decide whether the person is important enough to warrant disturbing a current meeting. The contexts utilised in the system are listed as; identity of users and visitors obtained through a question and answer process, the office owners schedule, the office owners status and the office owners willingness to see a visitor. This is obtained via interaction with the owner. Again the usage of the word context is different to any seen before.

The GUIDE system from Lancaster University is a prototype handheld tourist guide (Cheverst et al., 2000) which has been designed and deployed in real-world use throughout the city. The project has produced a great deal of pragmatic lessons for developers of context-aware applications. Their work is more appropriate to the pervasive computing world than the specifics of the work described in this thesis but the lessons are still worthy.

The GUIDE system uses 300 wireless communication cells in the city centre, each with a radius of approximately 300m. These disseminate information to the terminal as well as enabling the terminal to know its own location. The team list three prime goals of using context to improve the system. The main outcome is to simplify the input from the user required to adequately express their task or needs.

[1]Simple Object Access Protocol

A second is to simplify the choices and information presented to a user: for instance rearrange a list of nearby tourist attractions so that the open ones appear at the top of the list. The third is to reduce the size of the mental model the user needs to make a choice. This is achieved by an agent doing some of the work for them (Cheverst et al., 2000).

The lessons they present are as follows. The system must work predictably and consistently. They quote the 'principal of least astonishment' as the rule to follow. The system should improve the original functionality as opposed to confusing the user with a change in the mental model. There were drawbacks to the system. Using context to constrain presentation or the results that a system produces can be frustrating for a user. Their example is that the system could be used to list nearby attractions. The early versions omitted attractions that were closed in order to simplify the list. User feedback showed that this was frustrating some users who want to see a building just for its architecture alone. They therefore incorporated this into their model.

What is not acknowledged in their report is that this is a classic example of McCarthy's problem of generality, see Section 4.1. Here was a condition that broke their model forcing them to add additional complication into the system at a later date. This time they were lucky because the data model was unbroken in terms of the data stored about each building, they only needed to add to the user model a preference for seeing buildings from the outside. Their conclusion is that you should allow users to override the systems effects, a useful point to remember. They do not mention that such factors are always going to occur and it is impossible to cater for all possibilities. For instance, what if people wanted to see all buildings in the list to make plans for the following day when the locations would be open? Perhaps this fallibility is so obvious to them it does not warrant mentioning.

The quality of the work done in the project, including a comprehensive survey of user requirements, ensured that this type of minor alteration was all that was required. More problems were caused by difficulties with the physical implementation of the system as the devices had difficulty establishing their location with enough accuracy.

The pervasive computing community is tackling the problems of designing contextual systems to cope with problems such as change, distribution and reconfiguration. Whilst the AI community has many decades of experience in thinking through the underlying problems the pervasive computing world is taking steps towards pragmatic solutions to some of the problems.

4.5 Using Context

4.5.1 Hypertext Systems

There are numerous examples of the use of the word context in the hypermedia field. These tend to be highly pragmatic approaches and also tend to be unlike other perspectives already presented. A definition will be given for context which is often not discussed but just presented as a known quantity. Little acknowledgement is explicitly given to the wider debate on context as seen in the previous sections. Often context is another word for situation or a concept, it is another dimension across the hypermedia space to be utilised. Whilst the pragmatism and practicality of the approaches ensures that the authors can get on with building systems, there is little acknowledgement of other points of view on the subject.

A prime example stems from issues that arise when attempting to scale hypermedia systems to cope with very large amounts of data. In 'Dexter With Open Eyes' (Leggett & Schnase, 1994) discuss the notion of 'Hypermedia-in-the-large', drawing on the work of (Malcolm et al., 1991). Hypermedia-in-the-large examines the issues that arise when a hypermedia system starts to be deployed on a greater scale than was usually the case for most systems developed by research groups (Anderson, 2000). In their discussion they describe the need for a mechanism to reduce the size of the working set that the user is currently interested in, essentially a filtering mechanism to lower the number of links, documents and other components in use at one time. The notion was derived from examples such as Webs in Intermedia (Haan et al., 1992) and association sets in HB1 (Schnase et al., 1993).

The notion carried on into the work of the OHS Working Group. (Reich et al., 2000) defined Contexts as similar to these definitions. They cited (Delisle & Schwartz, 1987) who used context as a way to partition hypertexts into different views to aid collaborative working and deal with version control issues such as merging. The implementation view on the OHS system was that a Context object was a collection of id's to other objects in the systems, similar to the existing Composite object. The notion was not explored to much further extent within the OHS but individuals went on to produce new definitions in subsequent systems and models. There is little explicit definition of what context means in these papers which seems to hint that it is considered a known quantity.

Millard differentiates a Context from a Concept or a Composite object in the FOHM model (Millard et al., 2000) as implemented in Linky (Michaelides et al., 2001). The actual implementation of the FOHM model, the Linky service, uses Context as an extra mechanism for filtering results of queries to the link server. A Context object can be attached to various parts of the link model. The Context objects are implemented as a series of key-value pairs that may be matched against one another. Queries to the link server are made in context. Parts of the link structures that do not match the query context will be removed and the remaining link structures collapsed correctly. This allows for dynamic views on a single linkbase rather than choosing from a set of static linkbases.

The Dexter model was also extended by the Amsterdam Hypermedia Model (AHM) with the intention of including greater support for the complexities of presenting multimedia content (Hardman et al., 1994). Hypermedia linking, as opposed to hypertext linking, needs to take account of time-based presentations and linking into complex collections of dynamic contents, such as video being played alongside static images. The solution was the creation of link contexts, these are structures that give explicit instructions about which parts of a presentation are affected when a link is followed (Hardman et al., 1993). The work has primarily fed into multimedia presentation projects (Hardman et al., 1999) and has strongly influenced SMIL (Synchronised Multimedia Integration Language) (W3C Consortium, 1998b) and SMIL2 (W3C Consortium, 2001), languages for authoring interactive audiovisual presentations.

Hypermedia systems have taken advantage of contextual information for a long time, whether the word context was used to describe it or not. The primary area of work is the adaptive hypermedia world as discussed in Chapter 2. The adaptive user models and task modelling applications are contextually aware to a sophisticated degree though the methods for determining context and sensing change are limited.

4.5.2 Other Fields Using Context

An unrelated use of the word context is in information visualisation. Here the phrase 'focus+context' is a technical term for a system which visualises a large or complex amount of information by making a portion of it clear and detailed (focus) and displays a smaller scale or less clear version of the periphery of information (context) (Mukherjea & Hara, 1997).

An unexpected result of this investigation into uses of context is that the term 'contextual linking' is now used by the Web advertising industry. It defines advert links placed on sites that are deemed to be more relevant to the page requested by the user. The most common use is on search Web sites. The user enters a search term and the adverts placed on the returned page will be related to the query. The market is reaching maturity with there being a considerable level of sophistication in both the marketing techniques and technology being developed.

For instance a piece of software for creating the adverts for a travel Web sites called Tripadvisor[1] can provide packages of contextual advert links on a page. These links take into account the locations the user is interested in and the activity they are engaged in. For instance the user could be looking at hotels in a certain town. The software will provide links to a range of services for booking hotels in that town as well as fares for travel to the town. The links are categorised by the staff of the company for the customer. These technologies are advanced keyword matching systems, perhaps aided by some taxonomy or ontology work. The use of the word context is reasonable for the markets the companies work in but by the more stringent categorisation used here they are not strictly contextual. As there is no clear definition of context, their use of the word is as justified as anyone elses.

4.6 Summary

A context aware link service is one that would be able to cope with new situations, new document types, new link types and new methods of processing links. It should be able to change the underlying model used by the link service to use new input factors and new contexts. Understanding context has led to an investigation of its meaning and use in different fields of computer science. At the definition level there are multiple ways to describe its meaning ranging from the idea in which everything is a context to be measured and taken into account to the notion that context is one unmeasurable continuum derived from an unlimited set of factors. In order to progress towards a practical solution the notion of a measurable phenomenon is usually chosen by people wishing to incorporate context in computer systems. This model needs to be represented and the common lessons found amongst an array of approaches is that the modeller must decide what factors matter to them and what factors can be discounted from a model. This can be because the factor will stay static, and hence does not need explicit modelling, or that it is considered irrelevant or too difficult to measure.

Once a model is in place methods are needed to determine a change in the situation that leads to the adoption of a new context. Two relevant areas are explored, that of determining the subject of a document and pervasive computing. There has been considerable work done in systems that attempt to determine document subjects automatically. Such a technique was used in a previous thesis produced in the author's research group. Samhaa El Beltagy's work was similar to the authors in that it was concerned with providing generic links in context. However the major difference is that there her system used a fixed model of a linkbase in which a link has an attached context (subject) and there

[1]http://www.tripadvisor.com/

was a fixed method for determining the context. Her focus was more on the document understanding issues and the collaborative user aspects. This thesis aims for a more flexible approach to determining context and the ability to change how context is used.

The pervasive computing field has had some success in the implementation of systems to use sensors and systems to determine a context. The work is necessarily pragmatic but the models used can be sophisticated. Their major contribution is to make strong inroads into abstracting methods of detecting context changes and providing systems that can deal with new ways to determine context changes as they are required.

4.7 Discussion

This chapter started with the decision to build a new link service that provides links in context. But what is context? The daunting answer is that context is anything you want it to be. When considering the model to use in an application, if a factor can be considered to be static or not important then it may not need to be included in the model. But if we are to use links in applications that have not been decided yet how can there be a fixed model at all? As (Lieberman & Selker, 2000) says, "what you consider context depends on where you draw the boundary around the system you are considering". This does not even take into account the even harder problem that context is fluid and constantly changing. This derives from the definition of context as everything about a user that the software must take account of.

At the heart of the problem is that systems trying to take account of context suffer from the Frame Problem. The Frame Problem in the field of AI was first named by John McCarthy and is explained brilliantly by Daniel Dennett (Dennett, 1987). The problem is difficult enough for him to explain so my version will be brief. Essentially no computer programme's model can take account of all possible events or situations. The system model can be broken by circumstance or events considered outside the original frame of the problem. In our perspective any application designed to use context can always be found lacking in the way that it detects context shifts.

In a basic AI system a model of knowledge can be represented as a set of axioms and predicate logic is used to deduce the effects of the action. In the system a set of background axioms, called the frame axioms, would be used to describe general conditions and the general effects of actions in the system. The problem is that the frame axioms can always be broken by some unforeseen circumstance, hence the name.

For instance a robot is designed to navigate across a piece of land. It has a rugged all-terrain drive system and understands how to navigate boulders, sand and other obstacles. Perhaps it is being prepared to be sent to the moon. It is left in a desert with an objective to navigate to a point 50 miles away by itself. Suddenly it rains for the first time in ten years. As the desert is renowned for its dryness and the inventors are looking to send it to the moon the robot's model contains no mention of water at all. Especially streams. The robot drowns itself in the first one it tries to drive through. The designers have made an assumption that there would be no rain in the desert. They also assumed there would be no lava, deep snow or walking killer triffids and we instinctively know that they were right to do so. If they had heaped every circumstance they could think of into the model, no matter how unlikely, the robot would have ground to a halt whilst it iterated through all of them checking if the current circumstances matched. This seems to be something humans do not appear to have to do (Dennett, 1987). In each case a human would have naturally avoided the danger without needing specific warning or a re-write of his or her model of the world.

An even bigger problem is that of determining context changes. This chapter has primarily illustrated that anything and everything around us is a source of contextual information. Another constant theme is that there is no way to design an architecture that allows for all possible contextual applications and the sensing mechanisms they require. A compromise is always required. The pervasive computing world is leading the way with complex systems for abstracting the sensing of change and feeding that to applications. A possible reason for this success is that determining context changes is easier to do when physical sensors are used as opposed to attempting to use document understanding techniques. This work is focused on the considerable logistical issues involved in distributed computing and dealing with devices. The feeling must be that this area can only grow at a very rapid pace as the cost of devices, computing power and high bandwidth communications continues to fall. How many of the applications will succeed or be wanted by users is another question. The hardware technology may be outstripping the demand from users and the usefulness of the applications at the moment (Lueg, 2002).

4.8 Conclusion

The key lesson of this chapter is that the great thinkers tell us that models of context are implicitly fallible. They show that every model is going to be simplistic, breakable and often simply wrong. At the other end of the scale the pervasive computing community has shown great ways to incorporate sources of contextual information into applications and enormous progress is being made in taming the power of core technologies such as tiny devices, distributed systems and wireless communications. It is not surprising that much of their effort is taken up with the technical implementation challenges. However there does not seem to be a great deal of explicit acknowledgement that the models of context used will be flawed. There is little explicit planning for the time when the model breaks and some other factor has to be added. Whether it is a new attribute for an object, a new column to a database table or a change to the object API.

Trying to find advice on how to deal with the key issue of the context model breaking before it happens is very hard work. The vast majority do not acknowledge it or assert that their context model is sufficient. Most of the papers I have found sporting the keyword 'context' make no effort to acknowledge that their view on the subject is just one of millions. Most make statements to the effect that their system is contextual because of X implementation and that is sufficient justification.

This chapter has attempted to give a flavour for the many areas of research working on the problem of context in order to find some pragmatic advice for building a contextual link service. The conclusions have been daunting, the AI community has been finding context to be an extremely tough problem for decades. Overall it seems that attempting to define a model for some system, and declaring it to be a model for representing context, is a naive thing to do. Rather the safer option seems to work to the rule that there can be no single model for context, what is really needed is a way to work with as many models of specific contexts as necessary. This is the approach taken by the system described in this thesis.

The literature survey found that the term 'context' is used to describe attributes of systems that have no relation to each other. When writing a review of context-aware systems this causes problems. Perhaps the solution is to ban the use of the heavily overloaded word 'context' from descriptions of computer systems in favour of more accurate adjectives.

In the next chapter some of the lessons of this review are put to effect in an architecture for a contextual link service.

5 A Context-Aware Distributed Link Service

5.1 Introduction

The previous chapter illustrated that using the word 'contextual' when describing a computer system is a decision not to be taken lightly. Another key theme running through the chapter was that any application incorporating a model of context will be compromised (McCarthy, 1987). No single model of a system can successfully take account of all of the possible contextual situations that can occur. If the design acknowledges that perfection is impossible then much can still be achieved when writing contextual applications.

This could be construed as admitting failure before we even start a chapter describing a context-aware system. The answer is that such an attitude is healthy to take and avoids a temptation to make extravagant claims about the system in question. This leads to the suggestion that perhaps every system that claims to be context-aware should state a definition of context and what it is used for. More importantly, perhaps systems builders should acknowledge their own fallibility and state the limitations of their models and consequently what will break the system.

In Chapter Two existing open hypermedia link services were reviewed. The capabilities of the Southampton DLS were tested in Chapter Three when it was used with a document management system to provide a variety of linking services. The desire was stated to be able to provide new ways to determine the links to use in documents, such as by using the date of the document as a factor in the computation of the links. It was concluded that current link service architectures did not allow this. Existing systems were designed around one link model and one method of computing links. It was concluded that a system was required that allowed new models of links, along with a way to change the code that used the link model to compute the final links placed in documents.

This led to the idea of a contextual link service that could use new models of links and contexts to provide new linking applications. Chapter Four examined the meaning of context and its use in computer science. The conclusion from this examination is that modelling context is an unsolvable problem but that pragmatic solutions are possible. The important point is to know the limitations of any context model in use and when it becomes invalid to use it. This thesis advocates a system that can use many models of context in order tackle the problem of relying on a single model.

In this chapter, decisions are made on how to design and implement a working context-aware version of a Distributed Link Service. The process begins with a discussion of the problem of committing to a single model for the system design, for instance having one linkbase model or a single algorithm for generating links. The single model will always fail to adapt in some circumstance. Therefore a system is needed that can use multiple models. This chapter examines the architecture of a standard form of a DLS to understand what needs to be changed. The reference point for this work is the original DLS designed by Les Carr and Dave DeRoure (Carr et al., 1994). To implement the design change the notion of a Link Resolver is introduced. A Link Resolver encapsulates all of the elements of a DLS that will need to be interchanged as new models are required. The chapter goes on to describe the architecture of the new DLS incorporating Link Resolvers. In order to test the design, experiments need to be performed in which this new DLS is used with real world data. The following three chapters describe a number of uses for the core system in different experiments.

The underlying aim is not to write a link service to model context, but to write a link service to support multiple models for linking in context. The system is called the Context-Aware DLS, abbreviated to CA-DLS.

5.2 Design Considerations for a Context-Aware DLS

In order to design a Context-Aware DLS we first need to examine what functions a DLS performs and how previous applications of the technology have varied. See 2.3.2 for details of previous DLS systems.

Figure 5.1 shows the overall flow of events within a DLS such as that described in (Carr et al., 1994).

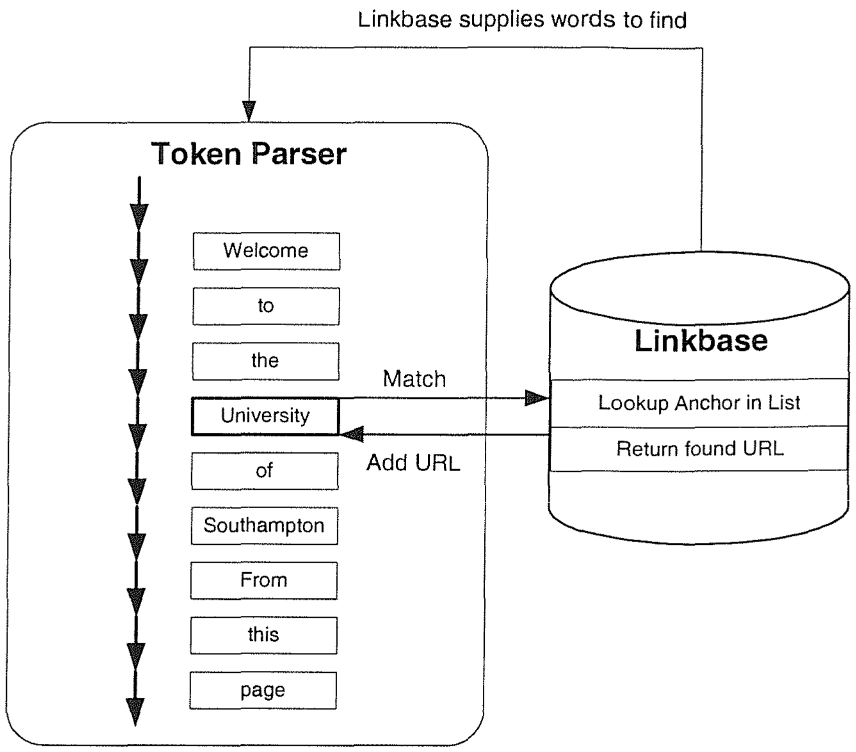


Figure 5.1: The Flow of Events Within a Standard Implementation of a Distributed Link Service.

The DLS parses Web pages into tokens and searches for tokens that are listed within the linkbase or linkbases. If a token is matched, the DLS will replace it with the HTML for a link anchor. The link destination is specified by the linkbase. The DLS will add links to every occurrence of the word in the Web page. The DLS can be written to provide alternative forms of links using different HTML. For instance underlined links or citation links placed at the bottom of the page (Hitchcock et al., 1998). The nature of the provided link can also vary. For instance an algorithm could be used to understand more about the subject of the page in order to provide better targetting of links. For instance an application could recognise citations or attempt to understand the subject of the page (El-Beltagy et al., 2001).

Each of the different versions of the DLS have been written as one complete implementation with a single model of computing and presenting links. In this thesis the aim is to develop a version of the DLS in which it is possible to replace the model in order to completely alter the system's behaviour.

The overall goal is to attempt to create a DLS which avoids the problem of generality (McCarthy, 1987), discussed in Section 4.1. This states that any particular model can always be broken by some unforeseen circumstance resulting in a rewrite of the system.

The first step to building such a system was to examine the general Carr/DeRoure DLS design and identify the components of a DLS that can remain constant and the components that need to be replaceable. From this a new DLS was designed which supports the interchanging of the key components in order to support new models of context.

The Context-Aware DLS is designed in two parts. The first part consists of the basic infrastructure including networking and document parsing capability. This provides all of the normal functionality up to the point when a word is identified as (potentially) being the source anchor for a link (i.e. it is matched against a term in the linkbase). It has no built in way to resolve the link and no set design for what form the link should take. These key functions are performed by interchangeable libraries. These libraries are called Link Resolvers.

For each method of providing contextual linking, there will be a new Link Resolver. If a new model of linking requires a new linkbase design, along with the code to utilise the design, then that can be written into a Link Resolver. If a method of linking requires access to other data sources, such as a database, then a Link Resolver can be written to perform such computations and plugged into the CA-DLS. In such a way new models for linking in context can be added into the CA-DLS without causing the underlying system to be rewritten.

Figure 5.2 shows the CA-DLS design incorporating Link Resolvers. A number of Link Resolvers have been written for this work. Each provides a different model for linking.

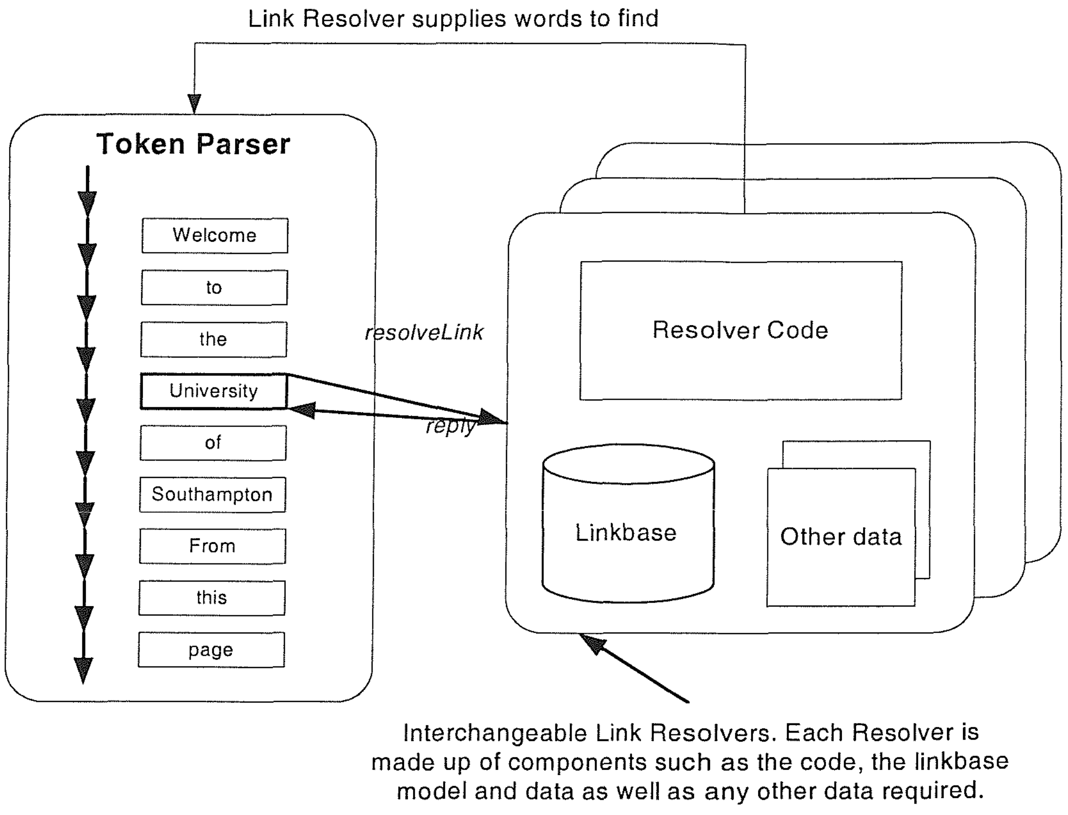

Link Resolver supplies words to find
Token Parser
Welcome
to
the
University
of
Southampton
From
this
page
resolveLink
reply
Resolver Code
Linkbase
Other data
Interchangeable Link Resolvers. Each Resolver is made up of components such as the code, the linkbase model and data as well as any other data required.

Figure 5.2: The CA-DLS Architecture Model.

In the CA-DLS the Link Resolver is responsible for the linking process. The main system will search for tokens in the documents and when it finds a match it will pass responsibility to the Link Resolver. The functioning of the Link Resolver is opaque to the CA-DLS and the two communicate through a simple set of function calls. The Link Resolver will generate and return HTML which will be placed into the document to form the anchor of a link.

The CA-DLS has also been implemented with the ability to dynamically change the Link Resolver during operation. It also has the ability to load and run Link Resolvers dynamically that were not compiled into the original system. This is discussed further in Section 5.4.2. This property gives further abilities to switch contextual processing models as required by an application of the CA-DLS.

The concept of the Link Resolver isolates the contextual modelling and processing components of the system from the rest of the infrastructure. This allows them to be replaced as new models of context are built or needed. This answers the first problem arising from Chapter 4 that the model can always be broken.

What has been designed has a similarity to a 'black box' approach. However the need to sense context changes and be open to new methods of context detection means the comparison is not quite true. A black box implies a sealed unit into which there are well defined inputs, processing occurs and an output is made. Designing an application to accommodate context means that the system cannot be isolated and inputs cannot be well defined. This is discussed brilliantly in (Lieberman & Selker, 2000), a context aware application must not only decide what to do based on the explicit input, but on the context as well. They consider context to be everything that affects the computation that is not the explicit input. Figure 5.3 shows how this affects the system design.

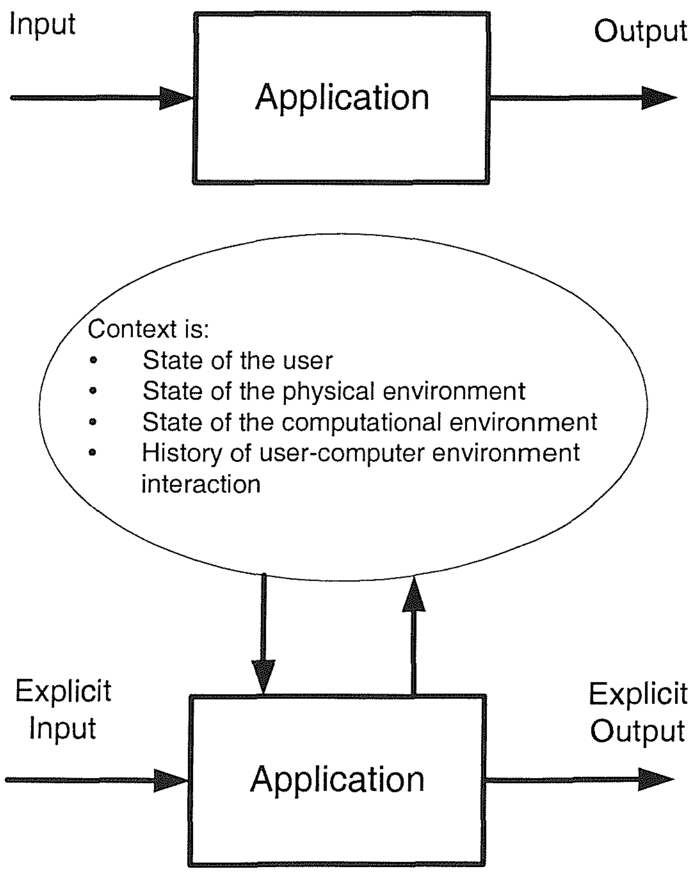

Input
Application
Output
Context is:
State of the user
State of the physical environment
State of the computational environment
History of user-computer environment interaction
Explicit Input
Application
Explicit Output

Figure 5.3: The Effect of Context on a Traditional Black Box System (Lieberman & Selker, 2000).

This returns us to the discussion in Section 4.3, how to decide where to define the model and draw the boundary around the model of the system. What is implicit and what is explicit? Again this leads to the frame problem discussed at the end of Chapter 4. Lieberman & Selker, (2000) highlight that scientists often deal with this problem by a process of 'reification', redefining the boundaries of the system so that what was external becomes internal. This leads to a need to be able to re-model the system, write new methods of defining and using context and a need for an open architecture for using multiple contextual models. With such a difficult problem to solve it could be said that the criteria for judging the overall system design a success can only be whether it can be reused in a number of different applications with a number of different contextual models.

5.3 Implementation of the Context-Aware DLS

In order to create the link service a search was undertaken to find existing technologies and components that could be used to build a proxy-based link service. It was reasoned that finding an existing Web proxy application and building the required components into it would save time. The focus of the work was not to build a proxy as this is a standard piece of software but to try and design a contextual link server. After a survey of available systems an application was found that fitted the requirements. The system chosen was called Muffin, a Web proxy with a modular architecture written in Java and described in Section 2.3.4.

The CA-DLS system is entirely implemented within one Muffin filter. The overall Muffin system was not affected. This new filter has an open architecture in which all of the processing of links and the model used to represent them is interchangeable. This Muffin filter is the core of the implementation work done for this thesis and is called the CA-DLSFilter. It is described in detail below.

The other major alternative to using Muffin for this work was WBI, described previously in Section 2.3.3. It is a larger scale piece of work used in commercial applications and has had a larger amount of effort put into it. The functionality is greater and the implementation more powerful and complicated. The work done in this PhD could have been implemented in this system but Muffin was chosen because of its simpler design and the example filters more closely matched the original project aims.

5.3.1 The CA-DLSFilter for Muffin

A new filter within Muffin was written to implement the CA-DLS. This CA-DLSFilter is the framework for Link Resolvers. It is described in detail in Section 5.4.

One of the sample filters included with Muffin is a simple Glossary filter. The Glossary filter implements the most basic form of a DLS. It searches for occurrences of a list of words and replaces them with links. It was deemed a promising framework for the requirements of this thesis and so was used to build the first version of the CA-DLSFilter. This was created while the author was working on the project described in Chapter 6.

The implementation of the core of a DLS is not a trivial task. The overall goal is to parse Web documents looking for words to annotate with links. The system quickly becomes one which has to deal with the many difficulties of parsing free text. Attempting to parse HTML adds to the complications. The parser must have an awareness of the structure of an HTML document and not attempt to add links within markup nor within sections such as the HEAD of the document. The code for such operations is time consuming to write and must be full of special exceptions and error handling. A large amount of the programming time for this thesis went into the parsing code.

5.4 Link Resolvers: Interchangeable Context-Aware Linking Engines

The architecture of the CA-DLSFilter is that previously seen in Figure 5.2. The CA-DLSFilter parses the HTML of the document looking for words to associate with link anchors. This list is supplied by the Link Resolver part of the system. When a potential anchor is found the CA-DLSFilter invokes a Link Resolver which generates the HTML to place in the document.

The name Resolver was chosen because a Link Resolver has the responsibility of resolving the form and destination of each link from the input information and other contextual inputs it requires. Each Resolver comprises a linkbase design and all the code to use that design plus any external data sources it will need. Each Link Resolver will load a linkbase specified in its preferences file. The CA-DLS does not provide the linkbase.

There is no set link model and each Link Resolver can represent and use links and other data as it sees fit. The code to load the linkbase and the code to use this data is unique to each Link Resolver. The Link Resolver library can be dynamically invoked by the system. A Link Resolver must conform to a certain specification which is detailed below. How it implements the specification is the responsibility of the Link Resolver. This simplicity allows freedom in the implementation of Link Resolvers and provides opportunities for contextual linking.

5.4.1 Link Resolver Supported Events

All Link Resolvers (shortened to Resolvers) must support four functions corresponding to four events in the system.

1. System Initialisation. *initializeResolver()*

When the system is started each Resolver is given the opportunity to load any data or links it requires. The Resolver must return a list of words which will be used by the DLS filter to search for in Web pages. In other words the Resolver must give a list of words to use as source anchors. How it generates this list is the sole domain of the Resolver. It can use a linkbase similar to that of other link servers or it can use any other data sources. The format and model of the linkbase used by a particular Resolver is not governed by any specification or single model. A Resolver can use whatever link model is most suitable to its particular functionality.

2. Adding a document header. *getHeader()*

When a document is requested the Resolver is asked to add extra HTML into the HEAD of a document. This allows the system to add in references to other resources such as JavaScript libraries or CSS stylesheet pages. An example of this can be seen in Appendix B, which is describing the details of an implementation of a Link Resolver application summarised in Chapter 6.

3. Adding a document footer. *getFooter()*

Similarly the Resolver is given the opportunity to add HTML to the end of the HTML document. This allows data or XML data islands to be embedded in the page if required.

4. Resolving a link. *resolveLink()*

This is the key activity of the Resolver. Each time a word has been identified by the document parser as representing an anchor, the Resolver is passed the word with other contextual information and asked to provide the HTML to place into the document. At this point it is the responsibility of the

Resolver to create a link or other HTML to place in the page. This is the core role of the Resolver and is how the separation is maintained between the basic system and the specialist components for implementing a contextual link service.

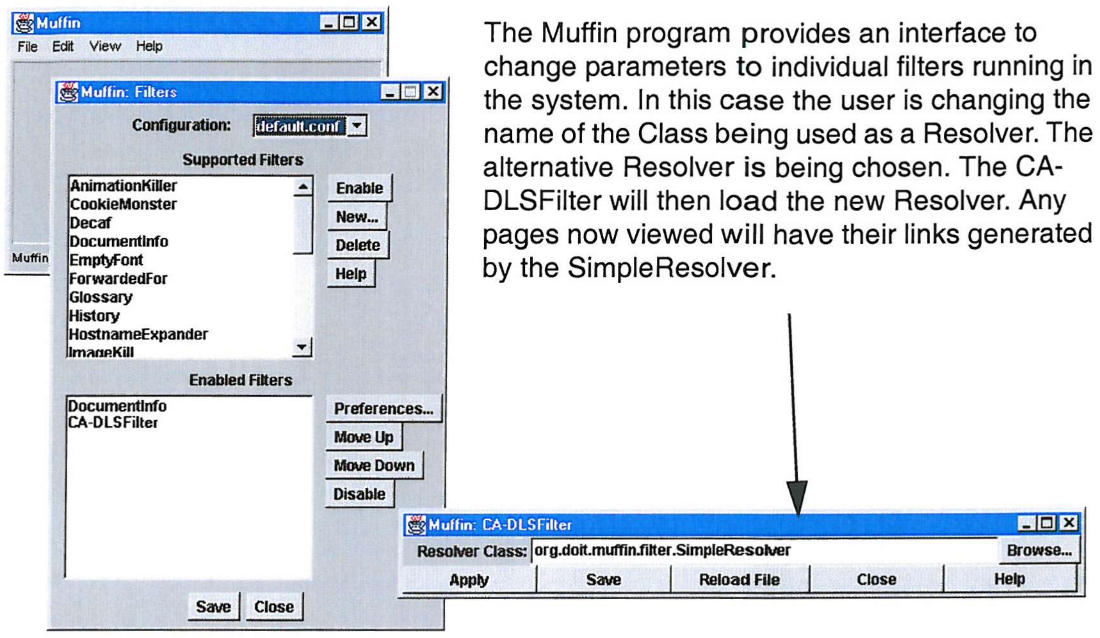
A crucial part of the design of the system is the interface between a Link Resolver and the CA-DLS. The goal was to allow Link Resolvers to be interchanged in a basic architecture and not to need to change any other code to significantly alter the functionality of the link service. Text is sent to the Resolver and text is returned. It is highly likely that the Resolver will require other contextual information in order to perform its function. The Resolver must be implemented to gain access to such data. For instance other data files might be required to supplement the data in the linkbase. The Link Resolver might also need access to real time environmental data about the system. This issue was discussed in Section 5.2.

5.4.2 Dynamic Link Resolver Invocation

The dynamic invocation of the Resolver means that it is possible to invoke a different Resolver each time a link request is made, even during the processing of a single document. In the case of adding generic links to a Web page, different Resolvers can be invoked for different words. It is also possible for the user to interact with the Muffin interface to choose a Resolver to invoke. This allows the user to radically alter the behaviour of the system whilst it is running. An example of this is shown in the next chapter which describes the first application of this system.

The invocation of the methods in a Link Resolver are achieved via the Reflection API of the Java programming language. The Reflection API represents the classes, methods and objects of a Java application. It allows a developer to obtain and use information about objects and classes whilst a program is running. It gives the ability to create an instance of a class whose name is not known until runtime and similarly invoke methods on that class.

The CA-DLSFilter is written in such a way that it does not know the name of the Resolver class until it needs it. The class name is not written into the source code. Instead it is given to the system as a parameter in the configuration file or from an interface control. In Figure 5.4 a user is entering the name of the class to use as a Resolver. When the 'Apply' button is pressed the system will load the class and invoke the *initialize()* method to allow the Resolver to load its linkbase and other data. This gives the user the opportunity to change the Resolver in use while the system is live. The example is taken from the system described in the next chapter.



Configuration: default.conf

Resolver Class: org.doit.muffin.filter.SimpleResolver

The Muffin program provides an interface to change parameters to individual filters running in the system. In this case the user is changing the name of the Class being used as a Resolver. The alternative Resolver is being chosen. The CA-DLSFilter will then load the new Resolver. Any pages now viewed will have their links generated by the SimpleResolver.

Figure 5.4: Changing a Link Resolver Whilst the CA-DLS is Running.

The system allows for more than one method of choosing a Resolver. The system is written to allow the code itself to choose which Resolver to invoke and it is possible to allow this to be done per link. For example a CA-DLSFilter could be written to choose between a number of Resolvers as it parsed a document. Different words could be linked in different ways.

The major reason for using the Reflection method of dynamically invoking Resolvers is to allow them to be added to a running system. It is possible to write a system where Resolvers could be found, installed and instantiated dynamically. The possibilities this introduces are discussed in Section 8.1.3.

5.5 Beyond Web Proxies and Link Filters

The architecture of the CA-DLS was designed so that the Resolvers are separated away from any of the Web proxy components. The Resolvers are deliberately encapsulated designs with the intention that the code could be used in other ways. The limitations of Web proxies for use as link servers were fully explored before this work began so it was always the case that a reliance should not be placed on them. For instance in Section 6.3 a Link Resolver is written and used within the Muffin proxy and also as a library to a normal program. Another prime example is the original motivation for this work, the integration of a link service with AIMS. The Resolvers could be directly run on the AIMS server allowing for the contextual link service to be directly part of the site. Unfortunately the design of Lotus Domino still makes it very difficult to access the content of documents from Java so the experiment has never been performed.

Another possible use for the system is complete contextual content generation. The system was originally designed as a way to generate fragments of HTML and place them in Web pages. The fragments of HTML were wrappers around link anchors to resolve the link to a destination. There is no particular reason why the system should just be used for making links. As the only requirement is that the Link Resolver should return a string, it could be used to generate pages that are specified by templates. Keywords or special markup would give instructions to the Resolver which it would use to perform some content generation and return a result. The result could be HTML, XML or any type of string.

In summary there are two overall problems, modelling context and detecting context changes. Of the two the first is the easier in this domain. It is a feasible proposition to find a way to encapsulate the design of a linkbase incorporating context, as this is enclosed deep within the system. However the places in the system where it is exposed to the outside world are much harder to deal with. The key problem is to design a mechanism for sensing context changes and communicating this information to the parts of the system that need it. This requires some assumptions to be made about how the context change can be represented in the system and communicated. The review, in Chapter Four, has shown that such things are impossible to do perfectly, for instance trying to convey information about a change in temperature in one circumstance but then being required to convey data about it suddenly raining. If the system does not know in advance all of the possible causes of a context change then there is bound to be a point when it cannot model some type of event and the design will fail. The frame problem applies to this design as much as any other.

5.6 Discussion

If there is some form of algorithm for creating or modifying links using some contextual reasoning then it needs parameters to use in its processing. These can originate from a variety of sources. They can be stored in the individual links. They can be obtained from the system or from user specified preferences. System or environmental data could be of the form of security permissions, operating system version or other basic information. User specified preferences, in the form of a profile, is the most common method of using contextual data. It is easy to extrapolate this list to include almost any measurable phenomenon. Chapter Four showed how context aware systems use such a wide spectrum of data, though not all at the same time.

The context review and the pervasive computing world have shown that the most difficult part in the design of any system incorporating context is the ability to detect a change in context. The application described in this chapter is weakest in this area. It is plainly obvious that a situation or use can be found for the system where it will be impossible for the Resolver to detect some phenomenon and hence not be able to determine the current context. It might be a limitation of the Java language to be unable to convey or represent some event. It might be a limitation imposed by the determination to make Link Resolvers a plug-in component sealed from the rest of the system. As we have seen the traditional black box approach to programming falls down when context must be considered because external effects must be taken into account.

This system like any other contextual system suffers from the frame problem. The point in the system in which the problem occurs is the function call that invokes a Link Resolver to resolve a link and return data. Whilst every attempt has been made to minimise the possible effects of the problem by making the constraints on the system as weak as possible this point in the design is still a flaw. It will always be a flaw as there is no possible solution that can work for all events and all possible situations. Therefore it is argued that the best that can be done is to openly acknowledge the frame problem and use all possible means to avoid it. Systems that do not claim to suffer from the frame problem but claim to be context aware are simply not being wholly truthful in the claims for their design. There is always a weakness which should be acknowledged and the bounds within which a system or design will not fail should be made explicit.

5.7 Conclusion

This chapter has presented a design for a Context-Aware Distributed Link Service. It is an evolution of the existing link services such as the Southampton DLS in two main areas. It is designed to support an unlimited number of link models and also an unlimited number of ways of computing the final link to place in a document. The system is designed with the intention that hypermedia application developers can build new links models and link applications to match their particular contextual requirements.

Rather than specifying what context is or declare a particular model the design acknowledges that any model of context to be used will eventually be shown to be flawed and so it is better to allow a system to use many models rather than just one. An open architecture for contextual link resolution has been described which attempts a solution to this. The system allows for the dynamic interchange of Link Resolvers. These are the core components that have total responsibility for how links are generated and are the location of the context models.

In the next two chapters examples of using the basic architecture are shown with a wide variety of applications and scenarios. Chapter 8, Conclusions and Future Work, looks at how well the design has worked and what could be done to improve it.

6 Applying the CA-DLS

The core CA-DLS design was presented in Chapter 5 as a new design for a contextual hypermedia link service. The CA-DLS is designed to be a starting point for writing a variety of context aware applications that are not limited to one context model or one linking model. In order to test the validity of the design the CA-DLS must be used to build contextual hypermedia applications. Only then can the design be evaluated as a realistic solution rather than a clever model. The writing and deploying of applications that use the CA-DLS will also help to explore the limits of the design in terms of the types of context models that can be utilised. This chapter presents the major uses of the CA-DLS and Chapter 7 explores a number of smaller experiments to examine the limits of CA-DLS.

The CA-DLS was built during the undertaking of two projects for an external sponsor, the CA-DLS creation being just one aspect of the work. This chapter describes the work done during those projects from a point of view concerned with how the CA-DLS was used. It summarises the other aspects of the project work emphasising the contextual linking approaches used. Appendices B and C are detailed accounts of the project work with information on exactly how the overall applications were built.

The CA-DLS was used as the core of a contextual link service delivering multi-destination generic links presenting the user with a weighting of the relationship between source and destination documents. The context models used in the CA-DLS Resolvers were based upon presenting links whose form and destination depended on which document, within a known set, the user was currently viewing. As the user viewed different documents within a document set the Resolver would present links which gave an indication of how relevant a particular destination document was to the current one for a given generic link anchor. The overall theme of the two projects was to develop methods to help users navigate through a large corpus of documents to find information that was useful and relevant to them.

6.1 Project Background

The work described in this chapter was carried out with the Post Office Research Group (PORG). PORG is a research division of the UK national postal service, recently renamed Consignia, possibly to be renamed back again. In 1997 a group of Consignia staff were involved in a contract to work on the privatisation and modernisation of the Argentine Post Office. These people participated in a knowledge capture and dissemination project at PORG because it was felt that their experiences would be of considerable importance to many parts of the organisation. This was known internally as the Argentina Knowledge cApture Project (AKAP). The main focus of AKAP was the development of a knowledge capture tool. The tool was designed to capture knowledge through a semi-structured interview process in which people were asked to recollect their experiences. People's names and commercially sensitive technical details have been changed for all figures in this thesis.

The result of the capturing amounted to a considerable number of documents, many of great length. A very large amount of time and effort went into the transcription of the documents from the tape-recorded interviews. When printed, the documents formed a pile of paper six inches high. The harsh reality was that AKAP had produced a document set so large and so verbose that no-one would have time to read them. If they contained important knowledge that could aid managers of the company, it was almost impossible to tell. Figure 6.2 shows a sample of one the documents and is typical of the content of the document set.

The AKAP researchers began two methods for analysing the document set. The first was to work on a Knowledge Visualisation tool with Dr. Chaomei Chen, then of Brunel University. This would combine various document analysis and visualisation techniques to discover relationships between the documents or between the keywords and represent it visually. The second effort was to examine the document set manually and attempt to extract the knowledge by hand. Finally the two approaches could be compared and contrasted. The results of these two efforts form the starting inputs for the two phases of work described here.

Chen's novel work (Chen, 1999) was in the merging of a statistical document analysis technique called Latent Semantic Analysis (LSA) with a technique for reducing the complexity of the resulting network of relationships, called Pathfinder. The result is visualised as a VRML[1] world.

LSA is a fully automated statistical technique which splits raw textual documents into words or phrases and builds a matrix of their occurrences in each document. Each entry is given a weight, either its weight in the context of the local document or a global weighting across the document set. From this matrix a variety of relationship matrices are mathematically derived; document-to-document similarity, word-to-document similarity and word-to-word similarity. The technique

[1]Virtual Reality Modelling Language

allows the user to manually intervene to reduce the word list. Stop words are automatically removed and all other words are stemmed. The user can then choose which words to use in the system and which to discard. In the case of the AKAP project Chaomei Chen chose the word list.

Even though the LSA technique already reduces the amount of relationship data produced it still produces such a complex network of relationships that it is of little use to a user. The Pathfinder network scaling algorithm essentially preserves only the most important relationships in the network using a more complex technique than simply deleting the lower values in the network. The result is a much simpler set of relationships called a Pathfinder network.

The Pathfinder network is then visualised into a VRML model by drawing it using force-directed graph-drawing algorithms. Each connection is treated as a hypothetical spring, with the spring strength relative to the connection strength. The model is then allowed to 'settle' into a minimum energy position. Strong connections will tend to be closer together and weak ones further apart. The user can view this network on VRML viewing software.

The end result of Chen's work was the delivery of a variety of VRML worlds showing various relationships within the AKAP documents. Chen delivered a variety of document-to-document networks, word-to-document networks and listings of the most relevant documents for each keyword. In Figure 6.1 an example of one of the document networks can be seen. The documents are mainly interview transcripts and are named for each interviewee. Each node represents a document and the clustering shows how the documents are related. In the VRML viewer each node can be clicked on to view the source document. Figure B.1 is another example.

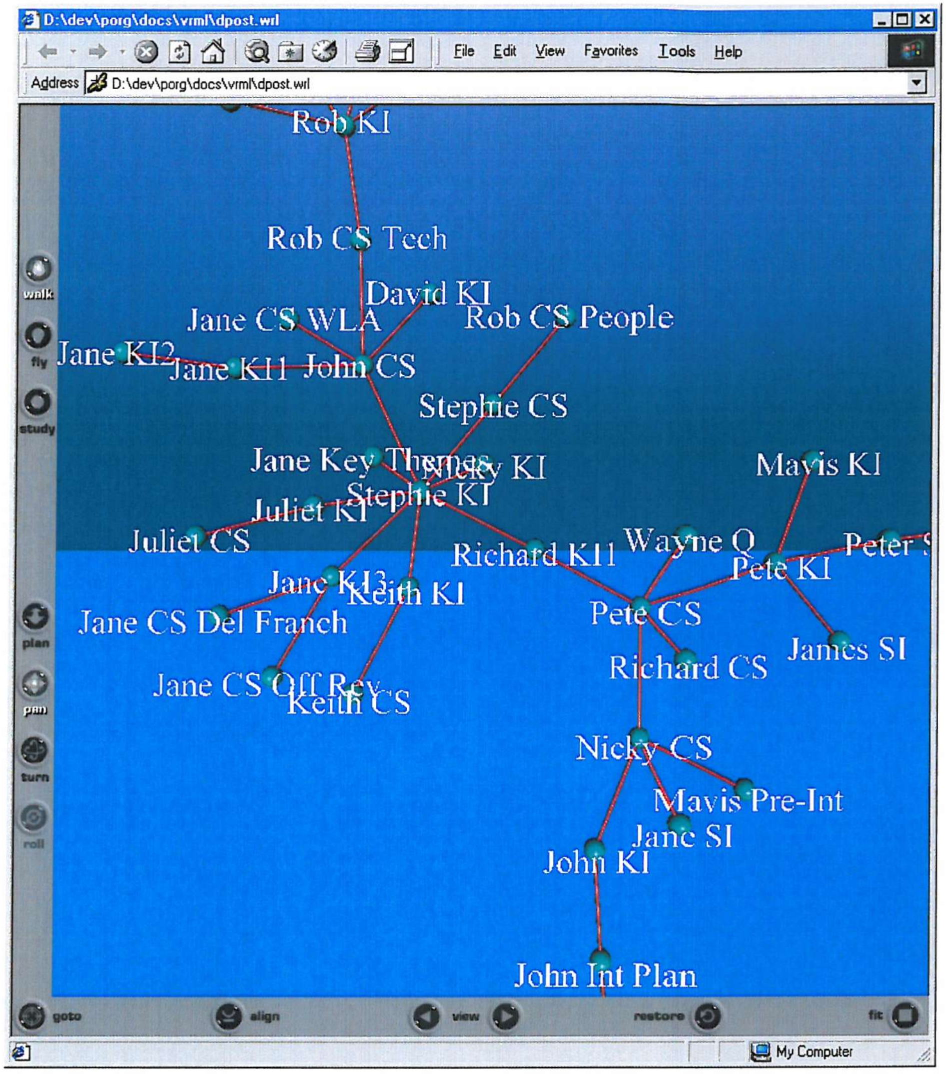


Figure 6.1: Example of a Pathfinder Network for Documents Shown in VRML.

It was felt by PORG that the visualisation process had been useful but there were usability issues. The VRML worlds were slow and difficult to navigate using the current computers of the time. An even bigger problem was the company ban on installing software not on an approved list. This included the VRML viewer and the latest version of Internet Explorer that was necessary to use the accompanying Web pages. The result was that the only people who saw the VRML worlds were the researchers on the project. None of the intended recipients could make use of the data. A more important problem was relating the 3D visualisations back to the documents themselves. In the VRML worlds of document-to-document relationships the user could click on a node and be taken to that document. In the keyword-to-keyword visualisations the user could click on a node and be taken to a listing of the 5 most important documents for that keyword. Once in a report there was little aid to actually navigating within the reports or clues on how to make use of the relationships that were being inferred.

At this point I was approached to attempt to build a link service that could make use of the data behind these networks and link into the documents in a more meaningful way. The first phase of work was to produce a system to add generic links into the AKAP document set that would utilise Chen's analysis work.

Once this phase of work was completed a second phase was proposed. Another, manual, analysis had been performed on the AKAP document set that listed paragraphs within documents that were felt to be the most important for a given topic. This data was used to build a new linkbase with a different model. An application was built to combine these paragraphs together to produce new documents that could be considered the highlights of the document set for a given topic. To do this the CA-DLS was used as a link service and queried from a new application rather than as a Muffin filter. The new application was a wrapper around an XSLT engine with the linkbase storing the transforms to perform on the document set. The final deliverable was a form of document compiler. The CA-DLS did not need any alteration to its design to be able to perform such a different operation from its original intended purpose.

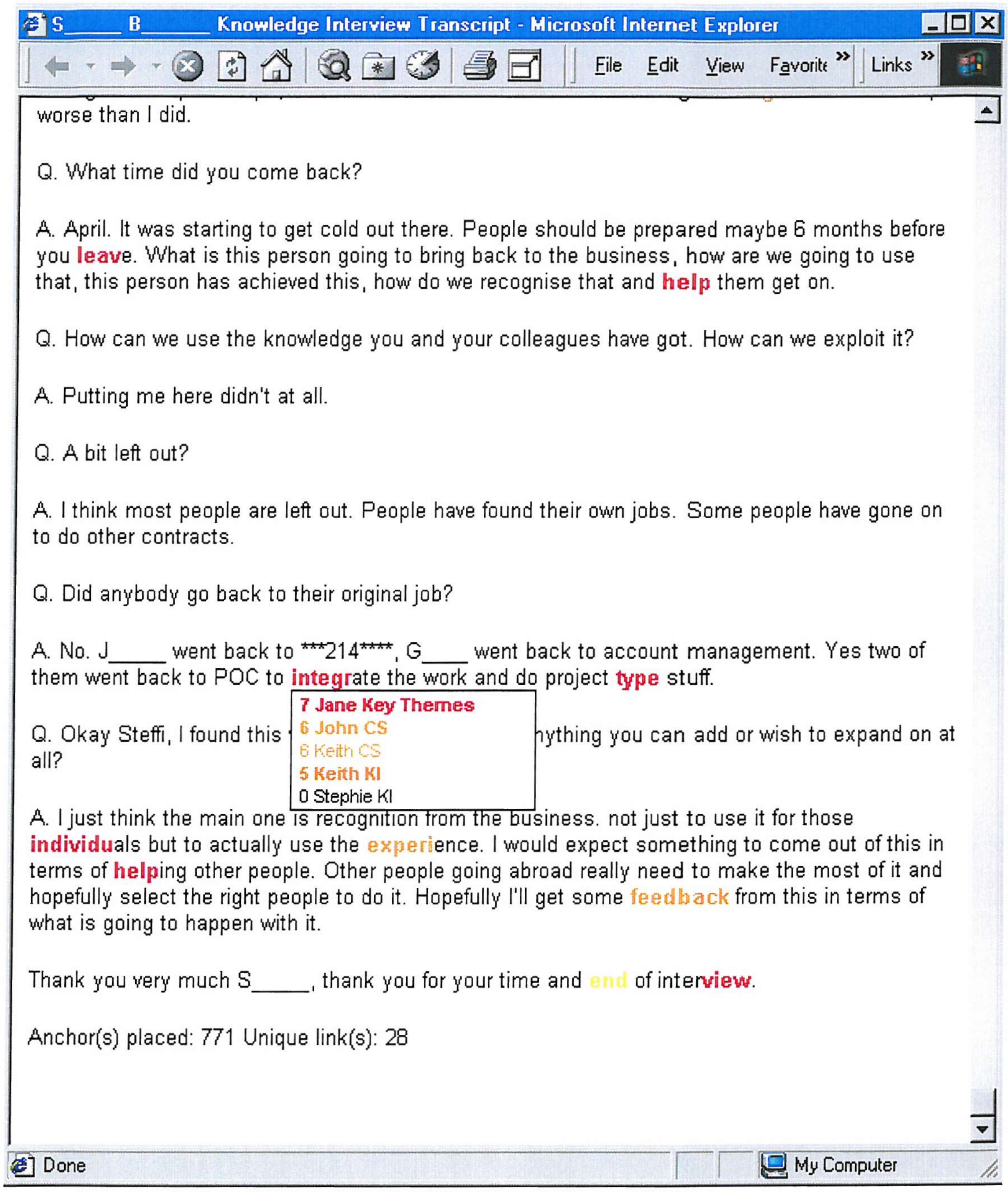
6.2 Phase One Implementation

The phase one system consisted of a web site of the AKAP documents complemented by an introductory page and contents listing. When a reader opened one of the documents using Muffin and the CA-DLS as their proxy the CA-DLS would add popup generic links to words in the documents. The link destinations were to other documents within the set. The destinations were chosen as locations of important information about the topic of the word being linked. Each link anchor had multiple destinations which were ranked and prioritised. This gave users a choice of documents to read on a particular topic and an indication of its relevancy and its relationship to the current document.

The system introduced a spacial context into the linking by ranking the links in a document depending on which document the user was viewing at the time. The documents in the system had been previously analysed and grouped in a 3D spacial viewing system. For instance documents that were considered to be more related were placed close together in the space. The link service used the data behind the visual analysis to give readers a feel for how related a destination document was as depicted in the 3D system. The links attempted to convey this by giving each destination a value between zero and ten and as well as ranking the multiple destinations for each link anchor. This was displayed using colours in the menus and a score between zero and ten; ten indicating the strongest relationship between the two documents at each end of a link.

The first Link Resolver for the CA-DLS was produced to add generic links to documents in which each link would produce a popup menu of five possible destinations. In this menu an indication was given of the relevance of the destination document to the current document by the use of a score for each link and ranking the links in the menu. Appendix B describes how the documents were processed and provides details of the HTML that the CA-DLS produced.

Figure 6.2 shows the system in action. Words that are coloured and marked in bold have been made into link anchors. A user has clicked on an anchor and a popup menu is presenting a choice of destination documents. Each is given a score which is also represented by the colour of the text.


S_____ B_______ Knowledge Interview Transcript - Microsoft Internet Explorer

File Edit View Favorite Links

worse than I did.

Q. What time did you come back?

A. April. It was starting to get cold out there. People should be prepared maybe 6 months before you **leave**. What is this person going to bring back to the business, how are we going to use that, this person has achieved this, how do we recognise that and **help** them get on.

Q. How can we use the knowledge you and your colleagues have got. How can we exploit it?

A. Putting me here didn't at all.

Q. A bit left out?

A. I think most people are left out. People have found their own jobs. Some people have gone on to do other contracts.

Q. Did anybody go back to their original job?

A. No. J_____ went back to ***214****, G____ went back to account management. Yes two of them went back to POC to **integr**ate the work and do project **type** stuff.

7 Jane Key Themes
6 John CS
6 Keith CS
5 Keith KI
0 Stephie KI

Q. Okay Steffi, I found this [illegible]ything you can add or wish to expand on at all?

A. I just think the main one is recognition from the business. not just to use it for those **individu**als but to actually use the **experi**ence. I would expect something to come out of this in terms of **help**ing other people. Other people going abroad really need to make the most of it and hopefully select the right people to do it. Hopefully I'll get some **feedback** from this in terms of what is going to happen with it.

Thank you very much S_____, thank you for your time and **end** of inter**view**.

Anchor(s) placed: 771 Unique link(s): 28

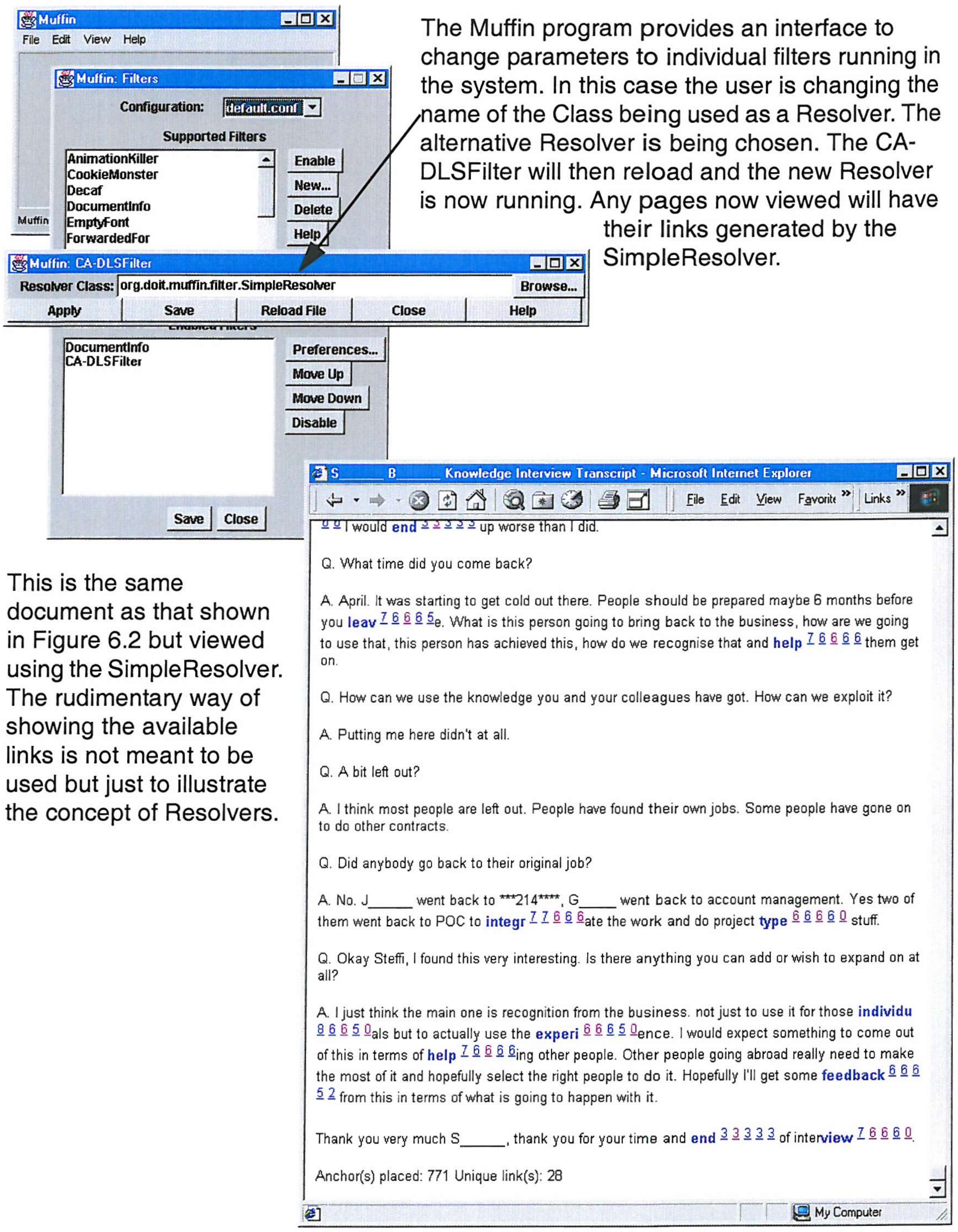
Done My Computer

Figure 6.2: A Document Enhanced with Multi-Destination Links.

The Link Resolver written for this project is described in detail in Appendix B. It used a linkbase written in XML using the XLink notation to describe the links. This data was supplemented with a set of data relating each document to the others in terms of a score between zero and one. The Resolver used this to generate the scoring of each link. The contextual input to the Resolver was the current document URL which was used to calculate document destination priorities. The Resolver was written to directly import the output data supplied by Chaomei Chen.

During the project an alternative version of the PorgResolver was written. There were two reasons for doing this. The first was that the end users of the system might not be able to make use of the popup menus written in JavaScript as they might not have the correct browser version. The second was to implement the ability of the CA-DLS to switch Link Resolvers whilst running.

The implementation was written to dynamically allow the user to change the Resolver in use whilst the system was running. A crude technique was implemented to facilitate this. In Figure 6.3 below the user enters the name of the class file of the Resolver and the CA-DLS instantiates the new Resolver immediately. The menu based links are replaced with in-line links. It should be noted that the processing and results are the same for both Resolvers as the development was just a proof of concept exercise. The key point of the exercise being to demonstrate the dynamic loading characteristic of the Link Resolver architecture.



Configuration: default.conf

AnimationKiller
CookieMonster
Decaf
DocumentInfo
EmptyFont
ForwardedFor

Resolver Class: org.doit.muffin.filter.SimpleResolver

DocumentInfo
CA-DLSFilter

S______ B______ Knowledge Interview Transcript - Microsoft Internet Explorer

0 0 I would **end** 3 3 3 3 3 up worse than I did.

Q. What time did you come back?

A. April. It was starting to get cold out there. People should be prepared maybe 6 months before you **leav** 7 6 6 6 5e. What is this person going to bring back to the business, how are we going to use that, this person has achieved this, how do we recognise that and **help** 7 6 6 6 6 them get on.

Q. How can we use the knowledge you and your colleagues have got. How can we exploit it?

A. Putting me here didn't at all.

Q. A bit left out?

A. I think most people are left out. People have found their own jobs. Some people have gone on to do other contracts.

Q. Did anybody go back to their original job?

A. No. J______ went back to ***214****, G_____ went back to account management. Yes two of them went back to POC to **integr** 7 7 6 6 6ate the work and do project **type** 6 6 6 6 0 stuff.

Q. Okay Steffi, I found this very interesting. Is there anything you can add or wish to expand on at all?

A. I just think the main one is recognition from the business. not just to use it for those **individu** 8 6 6 5 0als but to actually use the **experi** 6 6 6 5 0ence. I would expect something to come out of this in terms of **help** 7 6 6 6 6ing other people. Other people going abroad really need to make the most of it and hopefully select the right people to do it. Hopefully I'll get some **feedback** 6 6 6 5 2 from this in terms of what is going to happen with it.

Thank you very much S______, thank you for your time and **end** 3 3 3 3 3 of inter**view** 7 6 6 6 0.

Anchor(s) placed: 771 Unique link(s): 28

The Muffin program provides an interface to change parameters to individual filters running in the system. In this case the user is changing the name of the Class being used as a Resolver. The alternative Resolver is being chosen. The CA-DLSFilter will then reload and the new Resolver is now running. Any pages now viewed will have their links generated by the SimpleResolver.

This is the same document as that shown in Figure 6.2 but viewed using the SimpleResolver. The rudimentary way of showing the available links is not meant to be used but just to illustrate the concept of Resolvers.

Figure 6.3: Changing the Link Resolver in use With Muffin.

The document analysis work that fed into this project had delivered VRML worlds such as that in Figure 6.1. The external partners had found it difficult to derive a great deal of worth from them alone as they did not relate easily back to the original documents and there were problems with navigating the VRML worlds. The original reason for doing all of the work was to capture business knowledge from a group of people. Now that the source documents were enhanced with links based on the analysis data it became an easier task to examine them and question how well the knowledge capture process had worked.

Many of the documents were extremely long. In the 42 documents used in the work there were 284384 words amounting to 680 pages. It was an extremely difficult process for a reader to find the important pieces of knowledge from such an enormous amount of material. The bigger problem for the analysis projects that followed was that they relied on using keywords to describe the whole document. The underlying problem was the assumption that a long document can be described by five keywords.

Chen's analysis of the documents attempted to pick out the useful information from the surrounding 'noise' of the verbatim transcripts. The VRML work provided a start in this direction by providing means to navigate the document set and see the relationships amongst the text. The first phase of this project improved upon this by embedding those relationships back into the documents. This was hampered by the size of the documents.

6.3 Phase Two Implementation

Researchers at PORG had analysed the document set by hand in a laborious attempt to find the best information and summarise the findings. This involved physically reading all of the documents, finding the most important excerpts and forming an overall picture of the content. A document was produced, the 'Learning Summary' report, that contained an analysis of the documents and an in-depth summary of the knowledge found. The major contribution was a number of diagrams the authors called Knowledge Maps (KM) accompanied by references and indices. See Figure C.2 for an example. A KM is a hierarchical tree diagram of areas and sub-areas of knowledge covered in the document set. One KM, covering technical areas, is backed up by full references to the actual locations of relevant content on that topic. This analysis is of high quality and made good use of the documents.

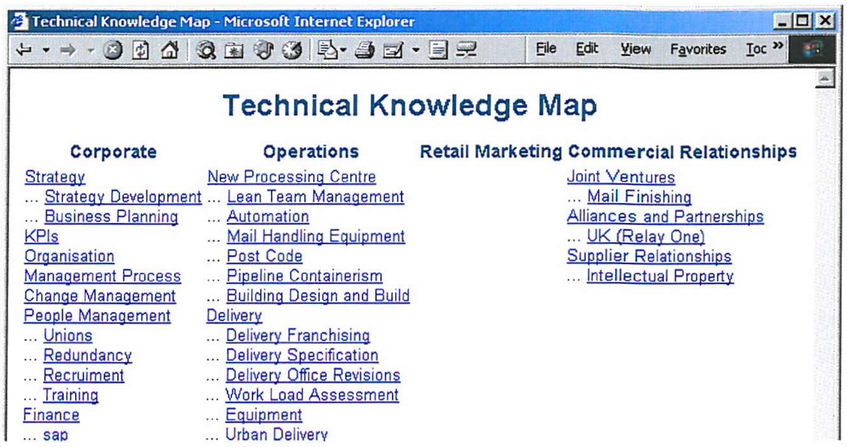
The diagrams were clear, well designed and formed a good overall understanding of the knowledge found. However the results were essentially 'locked' within this set of Microsoft Word diagrams and the data they contained was not reusable. The prime motivation of the second phase of work was to demonstrate that technologies such as open hypermedia and open standards could bring such work to life as well as enable knowledge reuse. On the implementation front the remit went beyond just extending the architecture of the CA-DLS.

The implementation consisted of reproducing 'Learning Summary' report as a living, dynamic Web site. Each part of the report was recreated using a variety of techniques to ensure that the knowledge could be reused and displayed as needed. The Knowledge Maps and references were re-written using XML and transformed to Web pages using XSLT transforms. This allowed for superior display techniques to allow users to view the KM's as well as allowing them to link into the documents or other data as required.

The 'Learning Summary' report was designed to accompany and reference the original AKAP documents described in the previous project. In order to use them with this project they were converted to XML, a difficult and time consuming process in itself.

During the project a second major Link Resolver was developed for processing a linkbase and activating XPointer links. The Resolver was utilised both using a standalone program and the existing CA-DLS. The standalone program was used to batch process the linkbase due to its size and the complexity of the XML processing being done.

The links in this project were from a structured diagram of areas of knowledge in the document set to individual paragraphs or pages in one of the documents. The source documents were the Knowledge Maps, in particular one Knowledge Map (KM) on technical issues in the document set. See Figure 6.4 for this Knowledge Map viewed on a Web browser. This KM was backed up by a number of pages of references mapping each technical issue to document file names and page numbers. This raw data was used as the start point for a link model. A summary of the model is included here, see Section C.6 for the full detail of the linkbase design.


Technical Knowledge Map - Microsoft Internet Explorer
File Edit View Favorites Too »

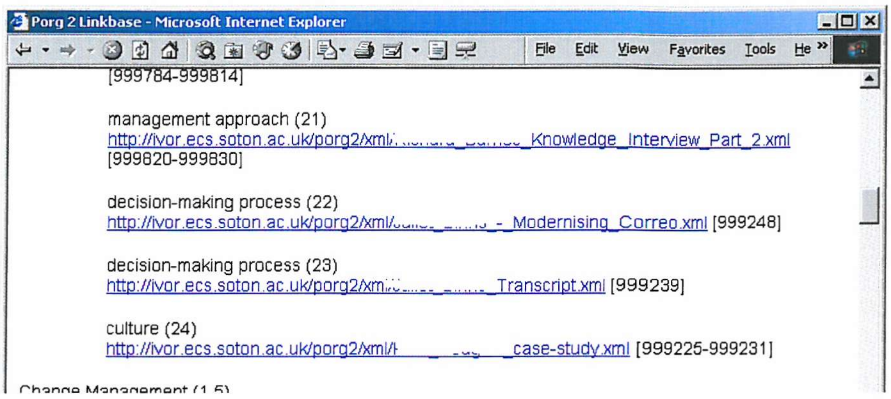
Technical Knowledge Map

Corporate	Operations	Retail Marketing	Commercial Relationships
Strategy	New Processing Centre		Joint Ventures
... Strategy Development	... Lean Team Management		... Mail Finishing
... Business Planning	... Automation		Alliances and Partnerships
KPIs	... Mail Handling Equipment		... UK (Relay One)
Organisation	... Post Code		Supplier Relationships
Management Process	... Pipeline Containerism		... Intellectual Property
Change Management	... Building Design and Build		
People Management	Delivery		
... Unions	... Delivery Franchising		
... Redundancy	... Delivery Specification		
... Recruiment	... Delivery Office Revisions		
... Training	... Work Load Assessment		
Finance	... Equipment		
... sap	... Urban Delivery		

Figure 6.4: A Sample of the Technical Knowledge Map Viewed in a Web Browser.

The technical KM represented the key pages in the document set for a set of technical subjects. See Figure C.8 for a sample of the XML. However these pages were not easily accessible to readers of the 'Learning Summary' report. From this starting point a system was built that could extract the paragraphs from the document set and compile them into new documents. These new documents would represent the best knowledge on each technical subject according the 'Learning Summary' authors. The new documents could then be given to interested readers targetting relevant information to specific people rather than relying on all users to search through the entire corpus of information for paragraphs of interest to them.

Figure 6.5 shows a sample of the linkbase as viewed on a Web browser, the XML can be seen in Figure C.9. The figure lists links against the anchor term, for instance the word 'culture', followed by the destination document and a range. The range, in square brackets, refers to paragraphs numbers in the XML. Some links are to a single paragraph and some are to a range of paragraphs. The XML representation of these links is produced using the XLink standard. Details of how the linkbase was designed and built can be found in Section C.6. In the figure the names of people have been blurred out for confidentiality reasons.



[999784-999814]

management approach (21)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_Knowledge_Interview_Part_2.xml
[999820-999830]

decision-making process (22)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible] - Modernising_Correo.xml [999248]

decision-making process (23)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_Transcript.xml [999239]

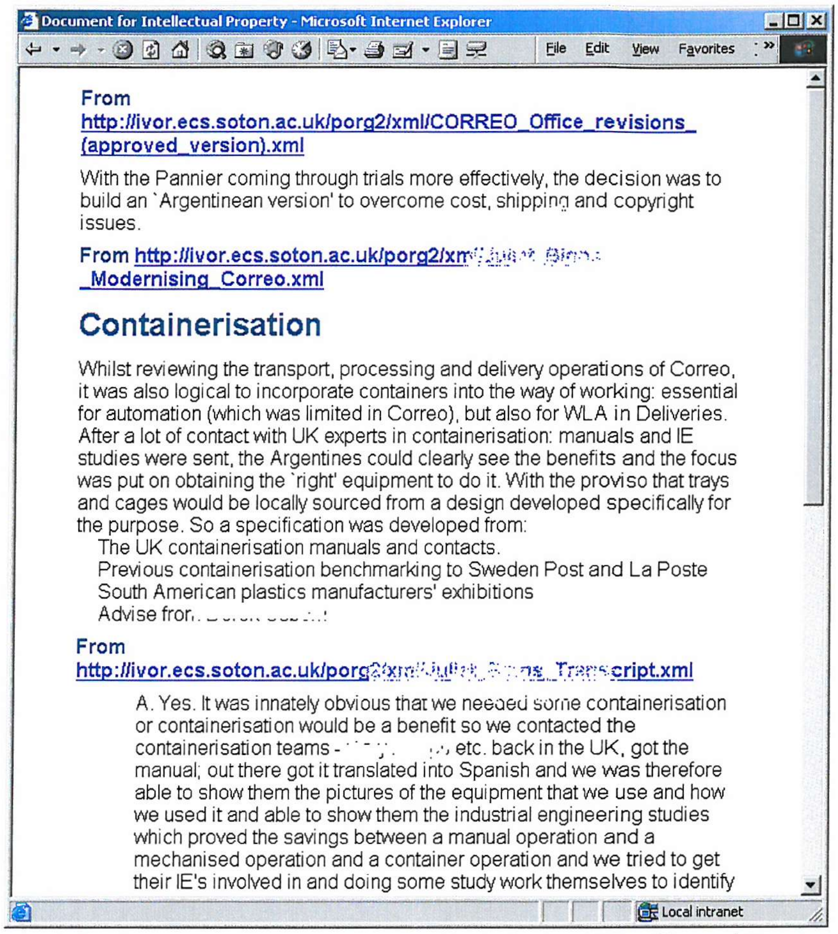
culture (24)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_case-study.xml [999225-999231]

Change Management (1.5)

Figure 6.5: A Sample of the Main Linkbase as Viewed in a Web Browser.

The finished application's purpose was to iterate through the KM and use each topic or sub-topic as a link anchor to query the linkbase. It would use the linkservice to extract the appropriate paragraphs and build a new document. Each link destination in the linkbase is an XSLT statement. The Link Resolver uses this statement with an XSLT engine to transform the XML original document. The result of the transform is just the selected paragraphs as XML. The application created a new XML document from a template and inserted each new XML paragraph into it.

Figure 6.6 shows one of the newly created documents. In this case the document covers the area of 'Intellectual Property' and includes paragraphs from a selection of documents in the set. Each included section starts with an explanation of its source and a link to that paragraph in the original source document. This allows users to follow a link and read the extracts in their original context. In the figure the names of people have been blurred out for confidentiality reasons.


Document for Intellectual Property - Microsoft Internet Explorer
File Edit View Favorites
Local intranet

From
http://ivor.ecs.soton.ac.uk/porg2/xml/CORREO_Office_revisions_(approved_version).xml

With the Pannier coming through trials more effectively, the decision was to build an `Argentinean version' to overcome cost, shipping and copyright issues.

From http://ivor.ecs.soton.ac.uk/porg2/xm[illegible]_Modernising_Correo.xml

Containerisation

Whilst reviewing the transport, processing and delivery operations of Correo, it was also logical to incorporate containers into the way of working: essential for automation (which was limited in Correo), but also for WLA in Deliveries. After a lot of contact with UK experts in containerisation: manuals and IE studies were sent, the Argentines could clearly see the benefits and the focus was put on obtaining the `right' equipment to do it. With the proviso that trays and cages would be locally sourced from a design developed specifically for the purpose. So a specification was developed from:

The UK containerisation manuals and contacts.
Previous containerisation benchmarking to Sweden Post and La Poste
South American plastics manufacturers' exhibitions
Advise fror[illegible]

From
http://ivor.ecs.soton.ac.uk/porg[illegible]cript.xml

A. Yes. It was innately obvious that we needed some containerisation or containerisation would be a benefit so we contacted the containerisation teams - [illegible] etc. back in the UK, got the manual; out there got it translated into Spanish and we was therefore able to show them the pictures of the equipment that we use and how we used it and able to show them the industrial engineering studies which proved the savings between a manual operation and a mechanised operation and a container operation and we tried to get their IE's involved in and doing some study work themselves to identify

Figure 6.6: The Generated Document for the Sub-Area 'Intellectual Property'.

6.4 Conclusion

This chapter has described the first full implementation project using the CA-DLS. During the projects described in this chapter the CA-DLS was designed and built. It was then used as a platform for building two differing hypermedia applications.

A proxy-based link service has been built to deliver multi-destination generic links presenting the user with a weighting of the relationship between source and destination documents. This weighting data was derived from previous analysis work of the document set by other researchers. The data maps to a 3D world in which the various documents or keywords can be visualised and the strength of the relationship between 2 documents or keywords is depicted by the distance between them. This maps to the links placed in documents giving users an indication of how related a destination document is, before they decide to follow the link. The CA-DLS also places links to the 5 'nearest neighbours', the documents deemed to be most related. The application helps users to find their way in a large document collection full of information that is not strictly important to them.

The context input to this system is the position of the current document within the 3D world and its relationships to those 'around' it. The data is provided by a number of source files and the entire processing system is self-contained within the Link Resolver. A second version of the Link Resolver was built that performed the same computation but returned a completely different style of links to the documents. In this case the links were normal HTML in-line links. This was to give users without the necessary modern browser the ability to still use the system. The implementation of this second version of the Resolver demonstrated the dynamic capabilities of the CA-DLS.

The effectiveness of the system for end users was affected by the quality of the documents supplied and the data supplied to build the links. The second phase of work effectively provided the reverse system. Rather than take the user to the paragraphs, the system compiled paragraphs with a similar subject, allowing improved navigation into the document set. Extracting paragraphs on a given topic and building new documents from them. The CA-DLS was utilised as the link service core of an application to provide linking from structures of knowledge stored as XML to the source documents themselves, again stored as XML. A new Link Resolver was written for the CA-DLS, called the XMLFragmentResolver, which understands XLinks. These links anchor one XML source to another and include notation understood by XSLT processing engines. The destinations of links are XSL transforms to apply to an XML document. The result is an extract from the XML file which can be returned by the Resolver.

The key contribution to the overall work is that this second phase of work for PORG saw the development of a much more powerful Link Resolver. This demonstrated how a Link Resolver can perform significant processing yet still fit into the CA-DLS design. The Link Resolver still conforms to the original specification but harnesses outside libraries, in this case an XSLT engine, and makes use of multiple sources of data. It demonstrates that Link Resolvers can be powerful systems in their own right and that the opportunity exists to interface with other systems in order to perform the contextual computation required for a particular application.

The CA-DLS implementation was further strengthened by the development of stable libraries and techniques for designing and using linkbases. It now became possible to rapidly design a new linkbase model and generate the code to load and utilise the model. The Link Resolvers described in the next chapter were developed very quickly due to this investment in library design.

In the second phase of the work the project moved beyond hypermedia and concentrated on XML processing and the power of XSL to extract data. The emphasis was on reusability of both data and software components as well as showcasing the activities of the W3C. For the project sponsors almost all the technologies and work described in this chapter were completely new. It is a disconcerting fact for a computer science researcher to realise that many computer users in large companies only know the applications they are given to work with. This invariably means the use of Microsoft Office. The work in this chapter is a significant distance from such tools and demonstrates that there is a considerable gap between the state of the art in computer science research and the harsh realities of computer users in large companies.

Now that the CA-DLS has been implemented attention can turn to examining the limits of the design. This chapter has described work in which the core system was implemented and used with real data. The context models themselves were simple in nature but the linkbase models were fairly complex to implement. Along with that there has been much implementation work to facilitate writing of new context models and linkbase parsers as well as numerous other infrastructure code required to produce the CA-DLS.

With much of the basic implementation work complete effort only needs to be spent on simpler experiments. These are ones in which context models and link models are designed or developed simply to test the ability of the CA-DLS to cope with such a design. Chapter 7 describes a number of new Link Resolvers and applications. In each case there are new forms of context model and hence new methods for detecting context changes. These contribute to the testing of the overall design and inform the discussion of how well this approach to writing a context-aware link service really works.

7 Examining the Limits of the CA-DLS Design

The previous two chapters chronicled the building of the CA-DLS and its use in two fairly different projects. Once these had been completed it was felt that, rather than build complete applications, smaller, more diverse, experiments should be carried out to test the design. To this end a number of Link Resolvers were written to perform some type of contextual linking but not fully deployed as finished applications. A number of further scenarios were developed for applications using the CA-DLS or the idea of the Link Resolver. These were used to consider the implications for the design and whether it would actually work if the systems were completed.

The weak point in the design was acknowledged in Section 5.6, it is the function call that invokes a Link Resolver to resolve a link and return data. There will always be some situation in which it is not possible to convey the necessary contextual inputs to the Link Resolver from the CA-DLS. This is not the same as data that the Link Resolver can obtain independently of the CA-DLS. If this happens then the only solution will be to redesign the system to allow for the representation and transmission of some new data type. This chapter is concerned with learning more about that limit.

The focus in this chapter is on the contextual link model within the Link Resolvers and what requirements that places on the overall system to detect the context changes involved in that design.

7.1 User Models for Resolving Links

As we have seen, the adaptive hypertext field is about building systems which model the user's goals, preferences and knowledge and use it to personalise a system. In this thesis we have, so far, concentrated on the infrastructure and methods needed to adapt links in context. This would form part of the infrastructure required to turn the CA-DLS into a fully fledged adaptive hypertext system but the system will also need to be able to model users. The Link Resolver developed in this section is a simple first step towards incorporating a user model and integrating with existing infrastructure to produce personalised links.

The Link Resolver described in this section provides links that are adapted to a type of user. The links are tailored to the position a person holds in an organisation and hence to the types of information that person is more likely to require. The Link Resolver has the ability to look up the person in a personnel database using a standard protocol called LDAP[1] (Yeong et al., 1993). With this ability the

Resolver can look up the position that a person occupies and hence determine a profile to use when resolving links. The Link Resolver, called LDAPResolver, runs within Muffin and the user would use Muffin as their Web proxy.

LDAP, lightweight directory access protocol, is an open networking protocol for accessing information stored in a Directory Services server (Bumbulis et al., 1993). Typically such servers hold personnel information such as names, locations, phone numbers and email addresses. One of the goals of writing this Resolver was to demonstrate how a Resolver could make use of the information already existing within organisations using open standards and protocols.

The Resolver holds a fixed list of people profiles, the directory services system is used to match an individual user to a particular profile. The linkbase model allows for links to be graded against each profile type. This allows the Resolver to score links based on how useful they might be to a particular user. The overall effect is similar to that in the first PORG Link Resolver from the previous chapter. An algorithm scores links depending on a context, in this case the context is determined by the username, and hence their user profile. The Resolver has been written to use a plug-in profiling system allowing for rapid development of new profiling applications or for the replacement of the existing scoring algorithm with a more complex one.

In this prototype there is no interface for the user to identify themselves to the Link Resolver. In a production system this could be achieved by the user entering their id into an interface component of Muffin or the user entering their id into the Web site using authentication or by the Resolver ascertaining the current user directly from the operating system. In the demonstrator Muffin is given a user id in the configuration file and uses that to query the LDAP server, all links are then adapted to suit that user type.

The LDAPResolver has a model which consists of an internal representation of the user's position within an organisation. It is a simple example of the type of profiles that could be built into this Resolver, these could take account of user information such as status, interests, age or project. Within the Department of Electronics and Computer Science a Link Resolver can make use of the internal personnel database containing information such as research group and whether the person is an undergraduate, researcher or academic. The simplest way to explain the nature of the database is that it is used to generate the department phone book.

[1]Lightweight Directory Access Protocol.

Both links and people are given scores between 1 and 10 in the areas of ‘research’, ‘academic’, ‘administration’, ‘teaching’ and ‘support’, the five major staff groups within the Department. When applied to a link these scores indicate how relevant the destination document is to those areas. The user profile works in a similar fashion. The user profile values are stored in a configuration file given to the Resolver. For instance all academics are given the same profile. When the Link Resolver adds links to a page the two sets of criteria are combined to score each link.

Table 7.1 shows an example scoring of a link against the profile of a researcher. The link’s profile suggests that the destination will be of most use to those who are more concerned with teaching. This is reflected in the overall score of 156 which is low compared to the possible total of 500. When the link is placed in the document the score is normalised to a value between 0 and 1, in this case the final link value is 0.312.

	Researcher Profile	Link Profile	Score
Research	10	2	20
Academic	5	8	40
Teaching	6	10	60
Administration	4	6	24
Support	3	4	12
Total			156
Normalised Value			0.312

Table 7.1: Example Scoring of a Link Against a Researcher Profile.

A link model has been developed to represent the profile information for each link. Figure 7.1 shows a simple example linkbase in which a generic link on the phrase ‘Courses Handbook’ has been defined to the online version of the Courses Handbook. The link has an Information Profile component ranking the link’s relevance out of ten on the areas of research, academic, teaching, administration and support. This link is most useful to academics within the department and administration personnel.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE link SYSTEM "status_linkbase.dtd">
<linkbase title="Test Linkbase with Status information">
<keyword
   href="http://localhost/foo"
   id="keyword_1"
   title="Courses Handbook"
/>
<destination
   href="https://www.ecs.soton.ac.uk/ug/handbook/"
   id="dest_1"
   title="Courses Handbook destination"
/>
<Information
   id="information_1"
   research="2"
   academic="8"
   teaching="10"
   administration="6"
   support="4"
/>
<go title="Courses Handbook Link" id="Link1">
   <from element="keyword" id="keyword_1"/>
   <to element="destination" id="dest_1"/>
   <attribute element="Information" id="information_1"/>
</go>
</linkbase>
```

Figure 7.1: A Sample Link for Use with the LDAPResolver.

When links are placed on the page the score can be indicated in the HTML of the anchor by some mechanism such as the colours described in Chapter 6. The exact calculation method and display mechanism are not important at this point and a new calculation method could be added to the Link Resolver in place of the one illustrated here.

The result is a Link Resolver that adapts generic links to the user of the system using a basic model based on the status of the user. The system determines the profile of the user by looking up status information from a live Directory Services system running in the department. The model is a simple one but designed to be altered easily to match new uses or more complex ways of using the technology. The CA-DLS architecture has not needed alteration to cope with the application and demonstrates potential for building a powerful adaptive hypertext system. The logical step would be to integrate the Link Resolver with an existing user profiling technology rather than create a new system. The use of the LDAP protocol is a demonstration of the ability to make use of external libraries from within a Link Resolver without affecting the overall system.

7.2 Links With a Context of Time

The context of time and dates has always been one of the key areas associated with this thesis. The initial problem which started the whole work on the CA-DLS was the flux of a workforce within an organisation and the need to reflect that in links for the AIMS system. In order to showcase this area a detailed plan has been produced for a Web site chronicling the story of the Millennium Dome with particular emphasis on showing how politicians change their views on a subject to suit the current political climate. The key action is to implement links with a timeline or plotline and be able to serve them effectively. This is described fully in Section 7.2.2.

First a Link Resolver is described which properly implements the links needed to work with AIMS as described in Section 3.3.5. It uses a variety of methods to determine the date of the document the user has requested. This then forms the basis of the Millennium Dome application.

7.2.1 The TimespanResolver

In this Link Resolver a model is used where each link has a start anchor, a destination and a time span. A time span consists of 2 dates forming a range. If the date of the document the user has requested is within the range of a link's time span then it will be added to the page. The objective is to develop applications in which a link service supplies appropriate links in context of the date of the document the user is reading. The date range components of the linkbase design have been added to the basic design and extra functionality written to allow a link service to query for links by date.

In order to work with dates and times in Java a general purpose class, TimeSpan, was written to interpret dates given in the linkbase. In the linkbase a Resolver author can specify the format of the date information given in the *start_time* and *end_time* components of a *TimeSpan*. The *Format* element contains text describing the layout of the dates and is given to the DateFormat class of the Java API. Figure 7.2 shows a sample of a linkbase incorporating time spans. A generic link is made from the acronym 'hod' to the home page of Professor Tony Hey. The link has a timespan which represents the time during which Professor Hey was Head of Department.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE link SYSTEM "timescales_linkbase.dtd">

<linkbase title="ECS Linkbase with Timescales">
<keyword
   href=""
   id="hod_ajgh"
   title="hod"
/>
<destination
   href="http://www.ecs.soton.ac.uk/~ajgh/"
   id="Tony_Hey"
   title="para_title"
/>
<TimeSpan
   id="1"
   format="d M y"
   start_time="23 February 1997"
   end_time="21 January 2002"
/>
<go title="HoD for Tony Hey" id="43">
   <from element="keyword" id="hod_ajgh"/>
   <to element="destination" id="Tony_Hey"/>
   <attribute element="TimeSpan" id="1"/>
</go>
</linkbase>
```

Figure 7.2: A Sample of a Linkbase Incorporating Time Spans.

The Link Resolver was written to find the date of a Web page by a number of means. It builds a cache of dates against URLs to speed link processing. If one method fails it tries others until it finds a date of some kind. Method 3 will always return a date though it may not be very useful. The methods are used in the following order:

1. Find a Date entry in the META tags of the HEAD of the HTML. This is where a date entry for the document is recommended to be placed in a HTML document. The system was written to be able to find the value of any META tag in HTML and then a date-specific function written to take advantage of that. The system will attempt a variety of ways to parse the text into a date and others can be given to the system as required by a specific example.
2. Find a Last-Modified date from a HTTP HEAD request. The HTTP HEAD request asks the Web server for basic information about a page. It may include a Last-Modified date entry which the Link Resolver will attempt to use.
3. Find a Last-Modified entry in the META data tags. In a similar fashion to the first method the Web page is parsed to find a Last-Modified date entry. This is a less reliable source of a true

date.

4. Methods were also written to find dates from specific archives and sites. For instance URLs to pages on The Times archive site contain a date within the URL itself. A method was written to extract them for the Resolver to use in a later example.

The AIMS system produces dates that will be found by the first technique and therefore this Link Resolver works easily with AIMS to find the date of a document. Now it is possible to correctly add date-specific links to AIMS, one of the original motivations for this PhD as described in Section 3.3.5.

In this case context detection is achieved simply by using the URL of the requested document. This is automatically given to the Resolver during the *resolveLink()* request so no further development was needed to obtain a context input. The Link Resolver can independently use HTTP and HTML parsing to obtain the information it needs. All data processing is done using the Date class from the Java API. It is a little more complex than is strictly necessary but once the work had been done to understand the Date Formatter class the work was reusable.

There are at least two applications of this technique with online archives as discussed in Section 3.3.5.

1 The simplest application is to place links in documents which match the date of the documents. Therefore if a link is valid for documents between 1998 and 1999 then it will only be returned if the document's date is within that range.

2 A more sophisticated example would be to allow the user to view an archive and be able to 'turn back the clock' through some interface. The archive and link service would show the state of the system at that time by only showing documents created up to that point and only showing links to match.

The concept of a Link Resolver that understands time spans can be expanded to more ambitious applications. The linkbase essentially forms a timeline of information. From here a possible avenue to explore emerges in the merging of work to produce a linkbase and work to produce a timeline. For instance history projects and news stories often feature timeline illustrations as a compact way to display a story summary. There is potential for the use of linkbases and time related link services here.

A related area is that of genealogy. There are standards for representing family trees in software. The GEDCOM[1] (Hawgood, 1998) standard is the underlying data format used by the majority of software in the genealogy industry. Such software is increasingly used to produce GEDCOM files of a family history which are uploaded to genealogy Web sites. It would be possible to write a Link Resolver to map from the genealogy data to links in the pages which correctly interprets the dates of people referenced in those pages.

7.2.2 Application Plan - The Story of the Millennium Dome

This section is an in-depth discussion of a possible application of this technology. It has not been implemented but a considerable amount of work was done in ascertaining the feasibility of such a design. For instance the archives of major online newspapers have been examined and the appropriate documents collected. The design is discussed in some detail before a discussion of the possibilities of how well it would work are considered.

The story of the Millennium Dome mainly spanned 5 years with a massive amount of media coverage. The many politicians involved changed their views on the project many times as the project was delayed, the vision was changed and site was eventually opened, run and then closed. The story is made more interesting because the project was initiated under the Conservative government of John Major and completed under the Labour government of Tony Blair. Politicians from both sides of the political spectrum found themselves having to change their public views on the subject when the change of government occurred.

Many people were involved in the project and many were casualties of the problems and considerable media criticism. Therefore there is a complex timeline of events involving a changing cast of characters and companies. Many online newspapers have good archives of the story along with analysis and historical overviews. When combined these provide the materials to build a new archive of stories complete with a timeline architecture and linking on key characters involved. The challenge is to show how contextual linking could add value to such a story.

The key aspect of the project would be to provide generic links on people and organisations which give a concise history of their involvement in the story. These links would be provided in the context of the date of the article being read. This first stage feature is interesting but of limited interest to

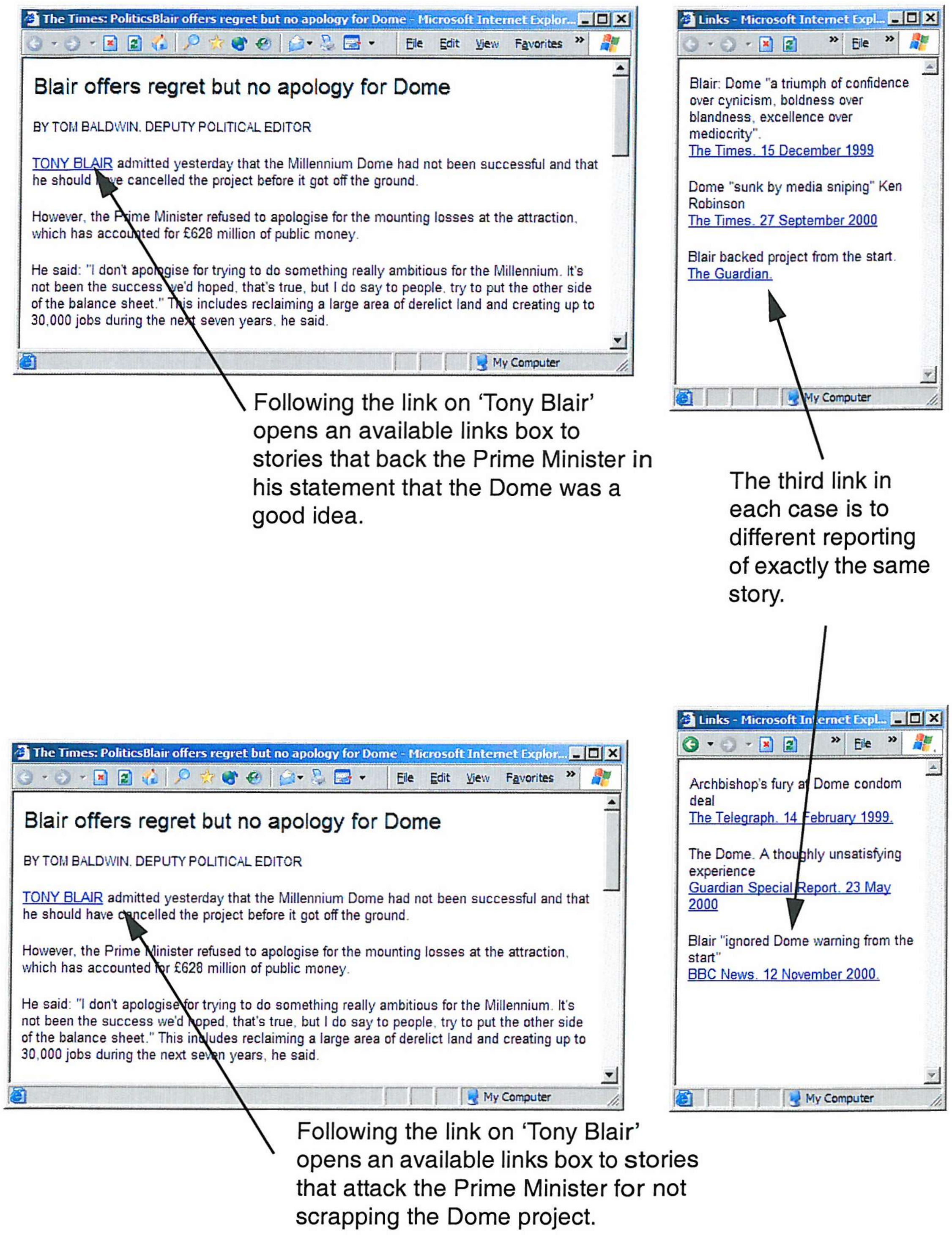
[1]Genealogy Electronic Data COMmunications

readers. The major showcase is to add value in the form of editorial opinion and analysis. This is also a form of context as differing views can be overlaid on the same corpus to provide radically differing views on a story.

A controversial showcase of this technology would be an application to highlight contradictions in people's statements. The reader would browse through stories augmented with generic links. Available generic links would normally include the most recent links or events involving the character such as recent statements or a list of key events they have been involved in. However an editor could decide to show links that illustrate how the politician is contradicting their previous statement or show how another person is backing their previous statements up and being consistent.

For instance if a reader clicks on a person's name a pop-up box could give a list of links to stories using the quotes of the person as the description. Which links are shown is up to an editor to decide. To add to the effect a reader could indicate preferences to the system and these could include their political sympathies. Therefore the whole system would be configured to give the user the version of events that appeals to their views. This would be a system that is not only using time as a context but is using the context of the editor's or readers political beliefs. A wholly subjective system. It could be argued that the context being used here is the context of 'spin'.

Figure 7.3 illustrates the potential effects of the system. The Link Resolver displays two different sets of destination links to related stories. It could be that the results depend on the preferences the user has registered with the system or that a site editor has registered with the system.


The Times: PoliticsBlair offers regret but no apology for Dome - Microsoft Internet Explor...
File Edit View Favorites
My Computer

Blair offers regret but no apology for Dome

BY TOM BALDWIN, DEPUTY POLITICAL EDITOR

TONY BLAIR admitted yesterday that the Millennium Dome had not been successful and that he should have cancelled the project before it got off the ground.

However, the Prime Minister refused to apologise for the mounting losses at the attraction, which has accounted for £628 million of public money.

He said: "I don't apologise for trying to do something really ambitious for the Millennium. It's not been the success we'd hoped, that's true, but I do say to people, try to put the other side of the balance sheet." This includes reclaiming a large area of derelict land and creating up to 30,000 jobs during the next seven years, he said.

Blair: Dome "a triumph of confidence over cynicism, boldness over blandness, excellence over mediocrity".
The Times, 15 December 1999

Dome "sunk by media sniping" Ken Robinson
The Times, 27 September 2000

Blair backed project from the start.
The Guardian.

Following the link on 'Tony Blair' opens an available links box to stories that back the Prime Minister in his statement that the Dome was a good idea.

The third link in each case is to different reporting of exactly the same story.

Archbishop's fury at Dome condom deal
The Telegraph, 14 February 1999.

The Dome. A thoughly unsatisfying experience
Guardian Special Report, 23 May 2000

Blair "ignored Dome warning from the start"
BBC News, 12 November 2000.

Following the link on 'Tony Blair' opens an available links box to stories that attack the Prime Minister for not scrapping the Dome project.

Figure 7.3: An Example of Editorial Linking in the Millenium Dome Application.

There are three layers of information in the system, these are shown in Figure 7.4.

At the bottom are the news articles. These stories would be gathered from standard news organisations and can be considered to be fairly objective. However all reporting is subjective to some extent. The advantage of an archive of stories drawn from multiple organisations is that many versions of the same event can be read by the reader helping to provide a more objective system.

The next layer is the collection of event-based links of major statements or events in the stories relating to characters and organisations. These would be written to be objective statements of fact.

The upper layer provides the way of choosing which links to provide to readers and is wholly subjective. It contains other metadata about the stories and links such as political bias and importance rating. This needs to be an extensible layer to allow reuse of lower layers of material for different purposes. This will even extend to deciding which versions of stories to present to readers. Some accounts of an event might be more sympathetic to a point of view than others.

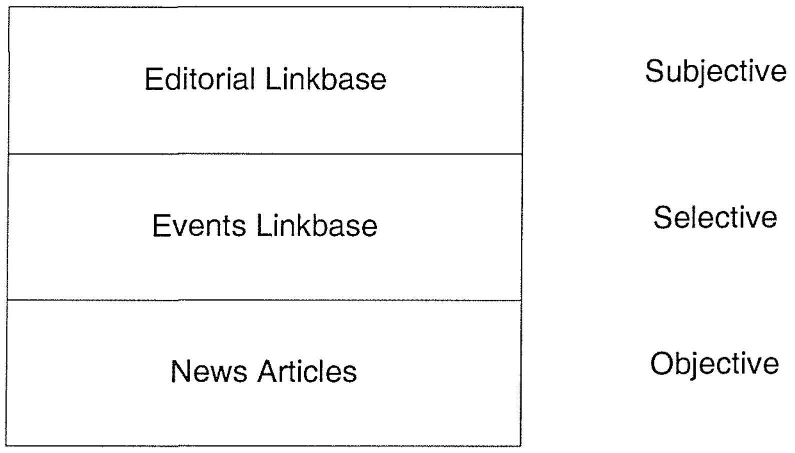


Figure 7.4: Data Model of the Millennium Dome Application.

From a linkbase design point of view there will be three major objects to describe in a link.

Stories. An interesting property is that stories are often about historical events. A news story may be about the emergence of facts about an event that happened earlier. For instance a leak to the media about what was said at a government cabinet meeting 6 months earlier. This requires careful thinking about the metadata used to represent the story.

Links to dated events. These could include simple summaries of the events. Links are primarily about an event. An event might be a complex object involving many characters and organisations used in the system. This would need careful designing to avoid overcomplexity.

Editorial links from stories to opinions. This is where the added value of the system is implemented in the form of editorial content.

In order to explore the implementation further a Link Resolver was written in which the following model of Link was built. The Link model took the form

Link = Entity + Story + Event + Information

This links an Entity, such as a person's name or an organisation, to an actual news Story, a time-based Event object and an additional Information entry. The Information entry would let a developer add arbitrary meta-data tags to the link. The Information object consists of an arbitrary set of key-value pairs which need to be listed within the linkbase, linkbase parser and the Resolver. This simple flexibility gives a powerful link model from which to begin the Dome system. A sample linkbase was built which matched this model and a parser also written to load such links.

To fully implement such a system would be a major piece of work. Not least in the creation of many links and the creation of the editorial linking. Story links would be relatively easy to harvest from news sites. Building Event links would require careful thought and almost certainly some type of timeline editor to aid in organising the chronological aspects to the data. There would also be considerable work in designing an underlying database to store the stories. The contexts involved would primarily relate to the date of the document or the simple identification of the document the user was reading. This would be in addition to a considerable amount of user profiling adaption needed to filter and shape links to suit the reader's tastes and political leaning. As the system would need to be based around an archive database it would not be difficult for the Link Resolver to obtain the metadata it required to help in link resolution.

The implementation would be an interesting challenge. There would be much to learn from the sophisticated news web sites at this point, many of which already implement a related links system. In order to fully integrate the system with an archive of news stories there would come a point at which it would be better if the link service was fully integrated with the web server so that pages were generated complete rather than the link service being implemented as a proxy or some other type of add-on to the site.

There is also a similarity to the work of Steve Hitchcock (Hitchcock & Hall, 2001) in which he has produced a form of journal portal. The system is a place for commentaries, views and discussion of papers. The papers are published elsewhere on the Web so the system forms a waypoint to discover and discuss these papers. He also uses a form of DLS linking to add links into externally stored full papers to add value or comments.

7.2.3 Generic Links for Works of Fiction Based on Timelines and Plotlines

This section briefly discusses a further application of time-based contextual linking.

When reading works of fiction it can sometimes be difficult to remember all of the information about the characters involved. It may be possible to use the Link Resolver idea to help a reader keep track of the characters. Links could be used with an electronic book reader to add annotation links about major characters by giving a quick biography of each character. The important feature is that the biography shown would only be the facts that the reader knows at that point in the story. The links would be displayed in the context of what 'page' the reader is on. When a user hovers the mouse over a character the link service would only return the fragments of the links that should be seen and hence give the right summary. A mode would also be required to return all the information at any point for readers who have finished the book, for example students studying the text.

The core of the implementation would be a system of links that form a time line for each character. This would essentially form a plot summary for each character. This would need to use parameters such as paragraph number or time as told in the story for its markers. Such an application would be closest to the system developed in Chapter 6 where links are related to paragraphs in documents using the unique paragraph number in the XML.

This system would not be a small task to implement, here we are more concerned with the implications for the CA-DLS design and whether it would break the Link Resolver model. The context detection system at the heart of this application would revolve around a technique for understanding the text the viewer is reading, this would map to the page in the original book. Paragraph markers such as those used in the second PORG application of Chapter 6 would be one method for solving this problem. This would not present great difficulties as an input to the Resolver. Therefore whilst such an application is far from easy to implement it would not create problems for the CA-DLS architecture. Therefore we can envisage that this is another contextual application that the CA-DLS could be used to produce.

7.3 Conclusion

Now that the CA-DLS is stable this chapter has demonstrated how a variety of contextual hypermedia applications can be created. The infrastructure for doing so is complete and there are now a number of reusable libraries for creating and manipulating link models in XLink. The solution to the original problem found in AIMS was solved quickly once the necessary date formatting library was written to accommodate links with a time span. Needless to say if a completely new link model is required that has nothing in common with XLink then that can be accommodated too. One of the key points of the CA-DLS design is that there is no need to have one single, complicated, link model which must always cope with all future possible needs. By hiding all responsibility for link models and link resolution inside an interchangeable component the application developer is free of many constraints. The problems of contextual systems have been avoided as far as possible by this design choice.

The Link Resolver design is very loosely constrained by the API it must conform to and that is reflected in the variety of applications shown here. The Link Resolver is so simply specified that there is no requirement for a Link Resolver to produce links. All a Link Resolver must do is return a string. Therefore a Link Resolver can be a simple wrapper around almost any existing functionality. This was demonstrated in the second phase of work in Chapter 6 when the XMLResolver returned XML paragraphs, sometimes of considerable size.

The range of contextual inputs to the system has been expanded so that the Resolvers have a large number of methods to ascertain the information they need to produce their results. Work has gone into writing methods for Resolvers to independently gather data from Web servers and Web pages such as dates and other meta data. In this case the Resolver needs the URL of the document the user is viewing, something that it has guaranteed access to from Muffin. The second major demonstration of this was to write a library to communicate with a Directory Services system using LDAP. This allows a Resolver an opportunity to query external resources for the information it requires. In this case the Resolver needs to be told who the user is by some means. This could come from an interface component of Muffin, data coming from Web browser authentication or even via the Resolver identifying the current user directly from the operating system.

All Resolvers require some seeding information with the *resolve* request or access to enough data to make their algorithms work. This is another way of looking at the edge of the capabilities of the CA-DLS design. The examples in this chapter were formulated to expand on the list of methods for obtaining the seed information to see how the design would cope. There will come a point where it will not be possible to write a Link Resolver that can determine enough information to use in its algorithm. At this point the CA-DLS will hit the frame problem and need a rewrite. That time has not come yet.

8 Future Work and Conclusion

8.1 Future Work

This section discusses areas of future work for developing the CA-DLS further and finishes with a list of implementation possibilities.

8.1.1 Detection of Context Changes

Roy Turner (Turner, 1997, Turner, 1998) describes a system, previously discussed in Section 4.3, for an autonomous robot submarine that attempts to use external inputs to match its current situation to a number of predefined contexts. Turner describes how this mechanism falls into three stages, recognition, tracking context change and changing the context knowledge. This thesis has not concentrated on the recognition or tracking areas but instead provided a framework in which such work can now be considered. The CA-DLS supports dynamic context switching as well as dynamic addition of new Link Resolvers to support new contexts. An important difference between the two systems is that Link Resolvers include executable code and can be more diverse in their operation. The robot contexts were all aimed towards helping the robot survive using changes in strategy.

An area for future work would be to concentrate on areas of context detection and mechanisms for context switching. This could be to use document understanding techniques such as those used by (El-Beltagy et al., 2001) or to look towards systems from the pervasive computing world. An investigation could be considered into utilising the Context Toolkit (Dey et al., 2001) with the CA-DLS as a framework for supporting a wide range of methods to switching contexts.

8.1.2 Transmission of Context Changes to a Link Resolver

The source of strength in the system design is also the weak point. It is the interface between Link Resolver and the main system through a single function call. This careful separation and definition of the interface between the main system and core functionality was the result of a lot of thought. It was also shaped by the experience of producing the diverse applications of the system. It gives considerable freedom to the implementation of Link Resolvers but is certainly not perfect.

There will be a circumstance in which the Link Resolver needs some piece of information about the current system, document or user status which it cannot obtain. To overcome this problem the main system will need to be rewritten to pass that particular information to the Link Resolver as a parameter in the function call. A solution would be to devise a method for the Resolver to describe its needs to the main system in such a way that the main system can automatically use that description to package the required data and send it to the Link Resolver. This was partially realised by the inclusion of a parameter to *resolveLink()* called 'Context', having the class of Object, the most general class in the Java language. The problem is to produce some form of shared understanding of what this Context object needs to contain so that the CA-DLS can supply it. The analogy is with the ontologies (Gruber, 1993) used by agent systems (Weal et al., 2001) or perhaps with the types of shared understanding being developed under the Semantic Web (Berners-Lee et al., 2001) work of the W3C.

For instance a Link Resolver might need to know the current location of processing within a document so that it can detect it is within the references section of a scientific paper and thus link citations properly. The solution might be of the form of a description file in a standard format that the Link Resolver designer can write and the CA-DLS can understand and act on when the Link Resolver is loaded.

As we have already seen there is no general purpose solution to this problem. This problem is another perspective on John McCarthy's problem of generality (McCarthy, 1987). He was talking about trying to build a general database of commonsense knowledge for use by any agent that requires it. This knowledge should be applicable in any context and not need modifying for any new purpose. He then identified that the real problem was finding a language to express general commonsense knowledge. I feel that finding a solution to such problems is outside the remit of this particular thesis.

8.1.3 Downloading Link Resolvers

A further stage to the work described here would be to implement the system as a personal proxy which makes use of published Link Resolvers downloaded from the Web. This section will examine how such a system could be implemented and evaluates whether it would be feasible.

One scenario for using the system is that a Web site or intranet could publish a matching Link Resolver for use with a personal copy of the CA-DLS. For instance the contextual processing to match the AIMS system is quite specific to that system and may not be meaningful to other sites. This introduces the notion that the publishing of a site should include the linkbase and the Link Resolver to match the site. The user's CA-DLS would find the Link Resolver, download and instantiate it as an accompaniment to using the site.

The user would run the CA-DLS as their personal proxy. The Link Resolver would be published as a download on the Web site. The CA-DLS could search for it via a special URL and then download the Link Resolver. This would include any other material the Link Resolver requires. Link Resolvers already have the ability to load linkbases and other data from URLs rather than just the file system so the idea is technically feasible. The Link Resolver would then be loaded as normal and given a chance to load any resources it required.

The single biggest objection to such a scheme is security. The very real problems of viruses on the Internet means that there is a culture of fear of downloaded applications. The target audience for most DLS applications has been corporate users of computers in which the links are related to their work. These users are the ones who are least likely to want to use this system or even be allowed to use such a system by system managers. In many companies installing software on computers without permission is a offence punishable by dismissal so the notion of this design would not be welcomed.

An interesting conclusion that can be derived at this point is that Link Resolvers and their context models are rarely reusable in their entirety. One of the key reasons for open hypermedia link services is reuse of data. This happens within applications as demonstrated by the power of generic links but more rarely at the higher application level. This is not the same for the code that has been developed. The Link Resolvers have been developed to be as general purpose as possible and easy to customise for new contexts. What has emerged is that the contexts need to match the applications closely and consequently are not so easily reusable. The context model is entwined in the particular link model and tailored to the content involved so this is not so surprising. It is not a great problem as good programming practice ensures that the components of a Link Resolver can be reused and remodelled as desired. This was demonstrated by the speed with which the Link Resolvers of Chapter 8 were developed.

8.1.4 Chaining Link Resolvers

A major issue with Link Resolvers is how to chain them together. As each Link Resolver is designed to be self contained it seems natural to want to put more than one in a system. For instance the user profiling work described in Section 7.1 would be a natural component to complement many of the other Resolvers described in this work. However the design does not easily allow for this as the Resolver is built to return completed HTML to the main parser of the CA-DLS. This is a weak point in the design and any future work should try to correct for this. Plenty of systems are designed in a modular way to form a chain and lessons can be learnt from them for a way to implement a solution.

The obvious one being the filter chain in Microcosm. The simple solution is to run more than one CA-DLSFilter in Muffin each hosting different Link Resolvers. This would keep the design clean and would achieve a great deal in terms of combining Resolvers. It also allows each CA-DLSFilter to look for different link anchors in the documents. The penalty would be an additional performance cost.

Another solution would be for the CA-DLS to call the next Link Resolver with the same parameters and supply the solution from the previous Link Resolver. This would be the HTML string (or whatever the string happens to be) that was generated to add to the Web page. This would require Resolvers to be able to parse and use that HTML in their own processing. In order to make this an effective system there would need to be tighter integration between Resolvers so that they would have a better awareness of what the other is generating.

8.1.5 News Archives

If a project such as the Millennium Dome archive was to be undertaken then it would be best linked to the work of Reuters in pushing towards a standard for representing multimedia news articles in XML.

There are two complementary XML standards for the representation of news articles. The first is NITF[1]. This is a schema for marking up textual news articles. It was started as an SGML DTD but is now based on XML. A more recent standard for representing multimedia news information is NewsML. This was advocated last year by Reuters. The International Press Telecommunications Council has now accepted the NewsML standard and a toolkit is already available for managing NewsML articles. The sample Web pages[2] indicate that NewsML could prove to be a valuable resource if such an application is envisaged. The drawbacks include the fact that there are at least 4 other standards for representing news articles; XMLNews-Meta, RSS 0.91, PRISM and RSS 1.0. It is unclear how these will evolve but Reuters have signalled their intentions to use NewsML and move their whole business operation to the Internet in the near future. Another problem with trying to build applications using one of these schema is availability of data. The system is designed for business to business communication and it is not clear that the raw XML feeds will be publicly available.

[1]News Industry Text Format
[2]available at http://newsshowcase.rtrlondon.co.uk/default.asp

8.1.6 The Semantic Web

The continuing development of the Web, and particularly the Semantic Web (Berners-Lee et al., 2001), has implications for the work presented here. In this thesis a vision is presented of an adaptive open hypermedia link service that uses multiple models and techniques to deliver links. The point of the thesis is not to focus on a particular model of context or one way to compute the links. The more interesting point is finding a mechanism to decide which context model to use and methods for detecting when to change context. As most of the contexts being discussed here are to do with documents it is now obvious that if this work were being started again today the tools and techniques of the semantic web would provide answers to some of the questions.

The semantic web is concerned with providing machine understandable documents on the web, as opposed to machine generated documents for human consumption. The prime goal is to provide structured semantic representation of the content of the document in a machine-readable way. This semantic representation will be domain-specific. This maps well to the notion of contexts in this thesis. For each domain ontologies are created that have certain bounds. They are well understood and match well to the data they are designed for. When moving between domains new ontologies and processing are required. In this work this maps to changing link resolver and link models. Therefore it seems logical that the way forward would be to use semantic web technologies to describe the context models and the capabilities, or domain, of a link resolver.

This issue was examined by (van Ossenbruggen et al., 2002), who used their Cuypers system (van Ossenbruggen et al., 2001) as an example of where semantic web technologies could be beneficial to them. Cuypers is a system for using high level descriptions of a multimedia presentation to dynamically deliver the final presentation to a user. It uses various models, such as the user profile, a platform profile, a domain model, design model and domain ontology as inputs to an engine. In this regard it has a similarity with the work presented here but is more closely related to the time-based systems described in Section 4.5.1. They embedded a variety of ontological markup within the SMIL presentations used by their system to examine their usefulness. This provided a way to describe the domain of the presentation to the system to use in adaption. A similar mechanism could be used by the CA-DLS as a mechanism for deciding which link resolver to use with a document.

8.1.7 Future Work Summary

1. *Link Resolver download and invocation.* A method for the CA-DLS to find, download and instantiate a Link Resolver and all of the supplementary components it requires. See

Section 8.1.3.

2. *Publishing a Link Resolver as part of a Web site.* A Web site could provide a Link Resolver to perform part of the linking between pages of the site. Users would then download the Link Resolver to their personal CA-DLS. See Section 8.1.3.

3. *Cataloguing and describing the capabilities of Link Resolvers.* A way of describing the functionality of a Link Resolver to enable reuse. This strays into ontology work and the Semantic Web initiative.

4. *Integrate pervasive computing systems that can detect and convey context changes.* The Context Toolkit (Salber et al., 1999) provides a framework for detecting and conveying events and sensor data. This could be a way to expand the capabilities of the CA-DLS to decide when to switch Link Resolvers. See Section 8.1.2.

5. *Utilise document understanding technology to detect context changes in documents.* Utilise technology to decide upon the subject of a document as a method to either pass to a Link Resolver as a contextual input or to trigger a change in Link Resolver. See Section 4.4.1.

6. *Investigate the usefulness of languages such as Prolog as a means to describe contexts and detect context changes.* This technique was seen in (Turner, 1998) see Section 8.1.1 and Section 4.2.

7. *Investigate methods for describing data that a Link Resolver will need to perform its functions.* Each Link Resolver needs certain external inputs and a more general purpose version of the system would need to involve some way to catalogue and describe the required inputs to the rest of the system.

8. *Chain Link Resolvers together at the filter level within Muffin.* A simple task would be to run more than one copy of the CA-DLSFilter in Muffin to utilise more than one Link Resolver. Each Link Resolver would add links to the complete document without any interaction with other Link Resolvers. A more powerful solution would be a system in which multiple Link Resolvers cooperate to produce links in one pass of the document. This would be much harder to implement as all of the possible interactions between Link Resolvers could not be known in advance. The solution would be to limit the scope of

Link Resolvers or to utilise a system such as that in the Microcosm filter chain in which a link is described in a message which all of the filters can alter and is only rendered at the end of the chain. Link Resolvers do more than just produce links so this would not be very effective. See Section 8.1.4.

8.2 Summary and Conclusion

This thesis has described the design, build and testing of an open hypermedia link service capable of supporting multiple models of contextual linking. The main objective was to investigate how context can be added into hypermedia systems in a general way and to avoid committing to a single model of context.

The history of open hypermedia link services presented in this thesis described the development of the DLS (Carr et al., 1994) from the link services in the earlier Microcosm (Fountain et al., 1990) system. The key idea propagated through these systems is that of a link from any occurrence of a word in any document to a single destination. Such links are called generic links and the standard usage is as a glossary or dictionary link in which a word is linked to its definition.

The DLS was used as the basis for experimenting with contextual linking by integrating it with a document management system. The nature of the content of the AIMS document management system allowed for many integration and linking opportunities. The system contained documents relating to the administration workings of the author's department and contained many references to people and other concepts within the department. Attempts were made to link these concepts with other online resources taking into account the context of the documents.

For instance, the historical nature of the system meant that documents contained references to people who had left the department. There were also links on acronyms for roles where it was not clear who it was meant to reference as the job had changed hands. The conclusion was that the link service would need to understand concepts such as the date of the document or some other attribute of the material. However each experiment found that the link service needed to understand and model some new attribute of the document, system or user and therefore needed different abilities for its particular version of context.

Various experiments in adding contextual linking to the DLS led to a set of requirements for a new link service in which multiple models of links with context could be used. A deeper investigation into what exactly context means and its use in computer science led to some challenging ideas arising from the work of the major thinkers in the development of artificial intelligence. Context is related to the frame problem (Dennett, 1987) in AI where a model is built for a system and the boundary conditions are set up to 'frame' the model. The problem is that circumstances can always be produced in which this frame can be broken.

The result of breaking the model of a system is that the system will need to be rebuilt in order to improve it. (Lieberman & Selker, 2000) calls this process reification. Any system that wants to avoid this problem cannot commit to a single model of context and therefore it was decided to design a system in which the model of context and link processing was written into an interchangeable component. The idea of a Link Resolver was introduced as a self contained subsystem which had total control of how links were modelled, computed and generated. The model of context and any code needed to work with this model was only held within the Link Resolver. The architecture provides the ability to change Link Resolver, and hence change contextual model, during operation in order to attempt to circumvent the frame problem.

The Link Resolver concept was introduced and the building of a Context Aware Distributed Link Service (CA-DLS) was described. The core functionality of the system is to act as a Web proxy service and parse documents as they are requested by a Web browser. The link service adds new links into the document. As the CA-DLS parses documents it looks for a set of words that are supplied by the Link Resolver. If one is found the Link Resolver is invoked with the word and other contextual parameters. The Link Resolver has the responsibility to return a string which will be placed into the Web page instead of the found word. The Link Resolver may use any model of linking and any model of context and any resources it wishes to compute the final value. The normal use is to return HTML containing a link anchor though there are a few restrictions.

The system was designed and built during the undertaking of two projects with an external partner. Both are described with attention paid to the model of links and the contexts they employ. The first project describes a link service that can add multi-destination generic links to Web documents. The links were also rated and ranked to indicate whether the destination document was closely related to the current document. The ranking data was the result of a previous project by the partner and had been represented using 3D VRML worlds. The documents were placed in the world so that related documents were nearer each other. Therefore the Link Resolver represented a spatial context allowing the user to relate the 3D visualisations back to the source documents and benefit from the analysis.

For the second project a different type of analysis work was used. A report had been written analysing the documents in considerable detail. This excellent body of work was reproduced on the Web using a variety of technologies such as XML and XLink linkbases. A key constituent of the report was a detailed knowledge map listing exact occurrences in the paper documents of useful technical information. This was reproduced as a complex linkbase that linked from each technical concept to the appropriate XML paragraphs. A Link Resolver was written to understand this linkbase and to harness an XSLT processor to extract the exact paragraphs from the reports. A program was written to use the Link Resolver to process the whole linkbase and build new documents from the paragraphs. The end result was a set of new documents containing just the high quality knowledge on each subject area listed in the report.

This system concentrated less on the context modelling aspects and more on the careful design of the system and the development of components to reuse in each new Link Resolver. The system also showed that Link Resolvers could be used independently of the CA-DLS in a standard application.

Following this, the limits of what could be achieved with the design were further explored. The tools previously developed were used to rapidly prototype two new Link Resolvers. Models of links based around contexts of dates and user profiles were built. A number of further scenarios were developed to the planning stage in order to examine whether the systems would be feasible and what the implications would be for the modelling of context and the detection of context changes.

The results of these evaluations of the system is that the design copes well with the hosting of alternative models of context. It was shown that there are few technical constraints on the processing that could be implemented by a Link Resolver as they can use any part of the Java language and make use of external libraries and facilities. The system has other significant abilities. It is possible to switch Link Resolvers dynamically, even to the point where a new Resolver can be invoked for every occurrence of an anchor in a document. It is also possible to start a new Link Resolver whilst the CA-DLS is running. A user could choose which Link Resolver to use or a system could be developed so that the main system chooses depending on some factor. This introduces a new level of context switching and, at first glance, is similar to the way that Microcosm and the DLS could swap linkbases. The DLS had the notion of a domain within which a linkbase could be processed, described by a partial URL address. That is a simple single context determination mechanism implemented in the main system whereas the Link Resolver idea allows for multiple methods of context determination. It would be simple to implement such behaviour in a Link Resolver.

A conclusion arising from the study of context is the idea that designers of systems that wish to support contextual behaviour by a single model should acknowledge that they are going to produce something fundamentally flawed. Any model of context within a computer programme will always be a simplification of reality and hence open to circumstances in which the model will fail. A suggestion arising from this thesis is that anyone wishing to claim that their system is contextual should declare their particular definition of context, what the domain is and what its limits are. More importantly an explicit statement of the circumstances in which the model would fail should be made.

In the CA-DLS the location in which the system will fail, the point at which the frame problem occurs, is in the interface between Link Resolver and the main system. There will be a circumstance in which the Link Resolver needs some piece of information about the current system, document or user status which it cannot obtain by any other means than the main system supplying it directly during the Link Resolver invocation.

The CA-DLS is a solution to the problem of rewriting link models and hypermedia systems to match new context models and new ways of generating links. It has been designed to allow for rapid and easy development of new contextual open hypermedia applications. This has been demonstrated by building a variety of applications in which the CA-DLS remains unchanged but the effects it produces are markedly different. The key point of this thesis was not to produce a single contextual hypermedia application. That has been done before, each with a new version of what context actually is and no agreement over what it really means. This thesis puts forward the claim that there is no definitive explanation of context, only application specific versions of a generally understandable concept. If there is no single, agreed definition then the solution must be to allow as many varieties as possible. The CA-DLS provides a platform for people to build contextual applications, each embodying their own definition of the word.

Appendix A:Overview of Relevant W3C Technologies

The projects described in Chapter 6 used a plethora of new and emerging standards for describing data and information. These standards are developed by the World Wide Web consortium, known as the W3C. The goal was to learn about and evaluate them in real-world projects for ease of programming, data representation and reusability. The following is a brief introduction to each technology to act a quick reference guide. Further information about each technology can be found from the W3C Web site www.w3.org.

A.1 A Beginners Guide to W3C Acronyms

This section begins with a simple summary of the acronyms used here.

Acronym	Introduction
HTML **(Hypertext Markup Language)**	The language used to describe Web pages. It is understood by a Web browser. It contains simple, fixed tags such as <H1> for a top level heading and <p> to denote the start of a new paragraph.
URI and URL **(Uniform Resource Identifier or Locator)**	A way of describing the location of a resource on a network. It provides a uniform syntax for describing diverse resources in an extensible manner. A URL refers to a subset of a URI describing the location of the resource and the primary access mechanism. An example of an access mechanism is the HTTP protocol.
CSS **(Cascading Style Sheets)**	A way to add more style to Web pages in a controlled manner. A simple language which defines how a Heading 1 should look, for example. Can be stored in a separate file from the HTML and so helps authors to manage their pages.
SGML (Standard Generalised Markup Language)	An ISO standard for a standard meta language in which all other markup languages may be expressed.
XML **(Extensible Markup Language)**	A way to represent any structured data in documents and on the Web. The basic building block of all new Web technologies. Unlike HTML there is no fixed definition of what any tag means, it is up to the applications to interpret the data as they see fit.

Table A.1: A Beginners Guide to Some Relevant W3C Technologies.

Acronym	**Introduction**
XHTML	The latest version of HTML written using the XML syntax.
XSL **(Extensible Stylesheet Language)**	Stylesheets for XML data. Much more powerful than CSS and much more complicated. Comprised of the following three parts:
XPath **(XML Path Language)**	Is a way to build an address path into an XML document in order to refer to a specific part of the data. Used by the transformation language to extract specific pieces of data.
XSLT **(XSL Transformations)**	Instructions that describe how to transform one set of XML data into another. An application can use the instructions to remodel the XML into something else. For instance turn XML to HTML.
XSL-FO **(XSL Formatting Objects)**	Formatting instructions and properties for presenting the transformed information.
XPointer	An improved version of XPath. It can additionally refer to a range in an XML file rather than just to a point.
XLink	A format for describing links between XML documents. In the language of this thesis it is a standard for describing linkbases.

Table A.1: A Beginners Guide to Some Relevant W3C Technologies.

A.2 XML

The core format used to represent the data and documents created in this thesis was XML (W3C Consortium, 1998a), a markup language for documents containing structured information. XML is a simplified descendant of SGML but is in fact an application profile or restricted form of SGML and can be read by SGML processors. The need for XML arose as the limitations of HTML began to hinder the development of the World Wide Web. HTML is a fixed markup format for the authoring of World Wide Web pages. For example in HTML the format and meaning of a top level heading is always a '<H1>' tag. By contrast XML is a generalised data representation language allowing the tags and semantics to be designed as required. XML is a generic data format expressly designed for use with the Web, for instance as a means to interchange data between Web based applications. The specification is compact but verbosity is common in files created with XML.

A.3 XSL

XSL (Clark & Deach, Aug 18, 1998) is a language for expressing stylesheets. It is the method with which the semantics of an XML document can be understood and the data used or transformed. XSL comprises 3 parts, XSLT, a language for transforming XML documents into other XML documents, XPath statements, for expressing locations of data within XML documents and XML Formatting Objects, a vocabulary for expressing formatting semantics.

A.4 XSLT

XSLT (Clark, 1999) is a language for transforming the vocabulary of XML documents into another XML vocabulary and is an inherent part of the XSL specification. It is designed to allow interoperability between applications and ease data exchange. XSL is a rendering vocabulary describing the semantics of formatting information for different media. XSL is primarily aimed at rendering XML to another format.

For instance a modern Web browser has an XSLT processor built in. If an XML document is opened in a browser and that XML document has the appropriate link to an XSL file to use then the browser can transform and format the XML document into a HTML document and display the results. The Microsoft Internet Explorer browser has a default XSL stylesheet capable of rendering an XML file to a readable tree for viewing in the browser.

A.5 XPath

XPath (Clark & DeRose (Eds), 1999) is a syntax for building an address path into an XML document and is used by XSLT and XPointer. The non-XML syntax allows a path expression to be built to address any location in an XML document from any other. The primary model of an XML document is of an arbitrary tree-like hierarchical model and the XPath syntax mirrors this. In fact it has a similarity to a URL in the way that it uses the "/" delimiter to dig down through the branches and leaves of an XML document to find the required node.

XPath has support for certain filtering expressions allowing an XPath expression to perform simple computations such as 'find the first "answer" child of the third "question" child of the current element'. However it is not a query language and has many limitations that compromise its ability. A major use of XSL files is to render XML data to HTML.

A.6 XPointer

XPointer (DeRose et al., 2001) can be characterised as 'XPath++'. XPointer is directly based on XPath but adds the abilities to address points and ranges as well as whole nodes. It is also designed to be used inside URI fragment identifiers. XPointer is designed for linking, it identifies sets of locations which are connected using XLink. XPath identifies nodes which can be used in an XSLT transformation.

In the project described in Chapter 6 a complex XLink linkbase was designed and populated that used 'XPointer-like' destination addresses. Unfortunately at the time of the project no software was available to support XPointers so the addresses really were a concatenation of a URL with an XPath statement. The link service utilised an XSL processor to process the XPath part of the address. The effect was the same as that which would be achieved by an XPointer processor and so the design was considered forward compatible. There is currently little software support for XPointer and the specification has been delayed for some time.

A.7 XLink

XLink (DeRose et al., 2000) is a language from the W3C for creating and describing links in XML documents. It provides a framework for simple uni-directional links and for complex, arbitrary link structures. The specification does not provide a design for a linkbase but a notation to use when representing a linkbase design in XML. It is up to the developer to design a linkbase to match their needs and to ensure that the linkbase can be parsed and used by an application.

There are two major types of links that can be modelled, simple links and extended links.

A simple link replicates the functionality of links in HTML. It is an outbound link comprising exactly two resources, a source and a destination. The syntax is different from that of complex links because they only require a subset of the functionality. They are of limited interest but prove useful for learning the syntax. Below is an example of a simple XLink.

```
<studentlink xlink:href="students/patjones62.xml">Pat Jones</studentlink>
```

Figure A.1: An Example of the Structure of a Simple XLink.

Extended links can have an arbitrary number of the following four elements in any order.

- **title** elements are human readable labels for the link.
- **locator** elements that address the remote resources participating in the link. For example the address of an external Web page.
- **arc** elements provide the traversal rules between the resources of the link. For example an element called 'go' which says that a link is followed from a symbol of a church on a map to a page about the church. It may also say how and when the link is activated.
- **resource** elements are local resources that participate in the link. These are essentially pieces of data stored as part of the link. For example the link to the church could have a resource element that is a label containing the name of the church.

Figure A.2 shows an extended link structure associating five remote resources. In this case it links five Web pages together concerning the different facets of the author's role within the department.

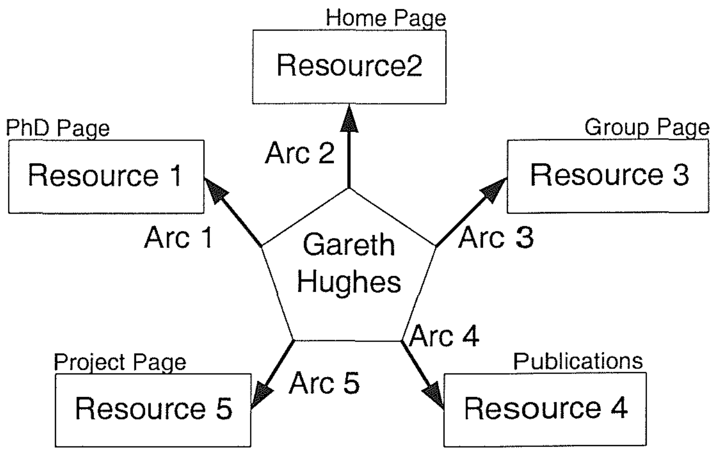


Figure A.2: An Example of an Extended XLink Structure.

The complex link allows a linkbase designer to create a single link structure joining together a plethora of resources whilst providing multiple ways to traverse between them.

Appendix B:Implementation of Chapter Six, Part One.

B.1 Introduction

This appendix describes a short term research project with an external partner. The project goals were written to allow me the opportunity to design and write the Context-Aware DLS. The project goal was to produce a link service for dynamically adding generic links to documents with multiple, ranked destinations. The work followed on from previous projects by the external partner and used documents and analysis results generated by other researchers as its starting point. It was hoped that a link service could add value to the documents by representing the analysis results in a usable way.

The final result was a link service providing generic links in the documents to other documents within the set. The destinations were chosen as locations of important information about the topic of the word being linked. Each link anchor had multiple destinations which were ranked and prioritised. This gave users a choice of documents to read on a particular topic and an indication of its relevancy and its relationship to the current document.

The system introduced a spacial context into the linking by ranking the links in a document depending on which document the user was viewing at the time. The documents in the system had been previously analysed and grouped in a 3D spacial viewing system. For instance documents close together in the space were considered to be more related than those far apart. The link service used the data behind the visual analysis to give readers a feel for how related a destination document was as depicted in the 3D system. The links attempted to convey this by giving each destination a value and colour as well as ranking the multiple destinations for each link anchor.

B.2 Background

This project and the one described in Appendix C were carried out for the Post Office Research Group (PORG), a research division of the UK national postal service, recently renamed Consignia, possibly to be renamed back again. In 1997 a group of Consignia staff were involved in a contract to work on the privatisation and modernisation of the Argentine Post Office. These people participated in a knowledge capture and dissemination project because it was felt that their experiences would be of considerable importance to many parts of the organisation. This was known internally as the Argentina Knowledge cApture Project (AKAP).

The main focus of the project was the development of a knowledge capture tool. The tool was designed to capture knowledge through a semi-structured interview process in which people were asked to recollect their experiences. Many were interviewed both before and after their time in Argentina. It was hoped to be able to identify what had been learnt during this time by identifying the differences in knowledge between the two document sets.

People's names and commercially sensitive technical details have been changed for all diagrams in this appendix and the next.

The result of the capturing amounted to a considerable number of documents, many of great length. A very large amount of time and effort went into the transcription of the documents from the tape recorded interviews. When printed, the documents formed a pile of paper six inches high. The harsh reality was that the project had produced a document set so large and so verbose that no-one would have time to read them. If they contained important facts that could aid managers of the company, it was almost impossible to tell.

The AKAP researchers began two methods for analysing the document set. The first was to work on a Knowledge Visualisation tool with Dr. Chaomei Chen, then of Brunel University. This would combine various document analysis and visualisation techniques to discover relationships between the documents or between the keywords and represent it visually. The second effort was to examine the document set manually and attempt to extract the knowledge by hand. Finally the two approaches could be compared and contrasted. The results of these two efforts form the starting inputs for the two projects described here and in the next appendix.

Chen's novel work (Chen, 1999) was in the merging of a statistical document analysis technique called Latent Semantic Analysis (LSA) with a technique for reducing the complexity of the resulting network of relationships, called Pathfinder. The result is visualised as a VRML[1] world.

LSA is a fully automated statistical technique which splits raw textual documents into words or phrases and builds a matrix of their occurrences in each document. Each entry is given a weight, either its weight in the context of the local document or a global weighting across the document set. From this matrix a variety of relationship matrices are mathematically derived; document-to-document similarity, word-to-document similarity and word-to-word similarity. The technique

[1]Virtual Reality Modelling Language

allows the user to manually intervene to reduce the word list. Stop words are automatically removed and all other words are stemmed. The user can then choose which words to use in the system and which to discard. In the case of the AKAP project Chaomei Chen choose the word list.

Even though the LSA technique already reduces the amount of relationship data produced it still produces such a complex network of relationships that it is of little use to a user. The Pathfinder network scaling algorithm essentially preserves only the most important relationships in the network using a more complex technique than simply deleting the lower values in the network. The result is a much simpler set of relationships called a Pathfinder network.

The Pathfinder network is then visualised into a VRML model by drawing it using force-directed graph-drawing algorithms. Each connection is treated as a hypothetical spring, with the spring strength relative to the connection strength. The model is then allowed to 'settle' into a minimum energy position. Strong connections will tend to be closer together and weak ones further apart. The user can view this network on VRML viewing software, a variety of which are freely available.

The end result of Chen's work was the delivery of a variety of VRML worlds showing various relationships within the AKAP documents. Chen delivered a variety of document-to-document networks, word-to-document networks and listings of the most relevant documents for each keyword. Figure B.1 is an example of the VRML worlds, this one is a network of all the keywords.

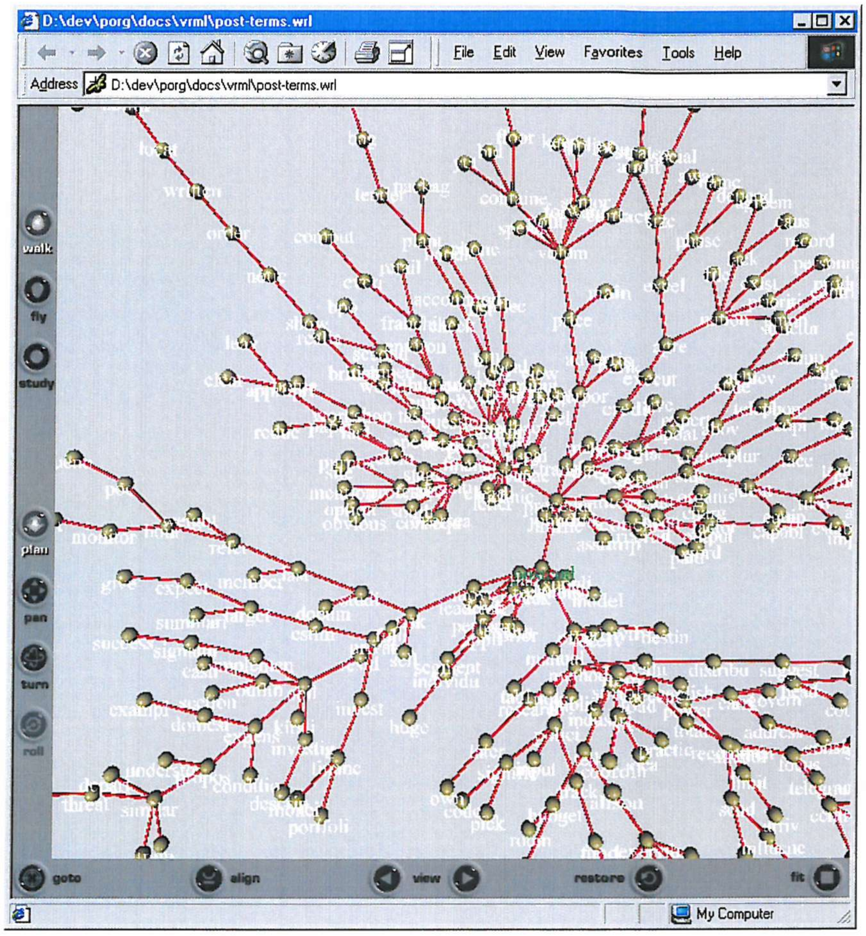

D:\dev\porg\docs\vrml\post-terms.wrl
File Edit View Favorites Tools Help
Address D:\dev\porg\docs\vrml\post-terms.wrl
walk
fly
study
plan
pan
turn
roll
goto
align
view
restore
fit
My Computer

Figure B.1: Example of a Pathfinder Network for Keywords Shown in VRML.

In Figure B.2 an example of one of the document networks can be seen. Each node represents a document and the clustering shows how the documents are related. In the VRML view each node can be clicked on to show the source document.

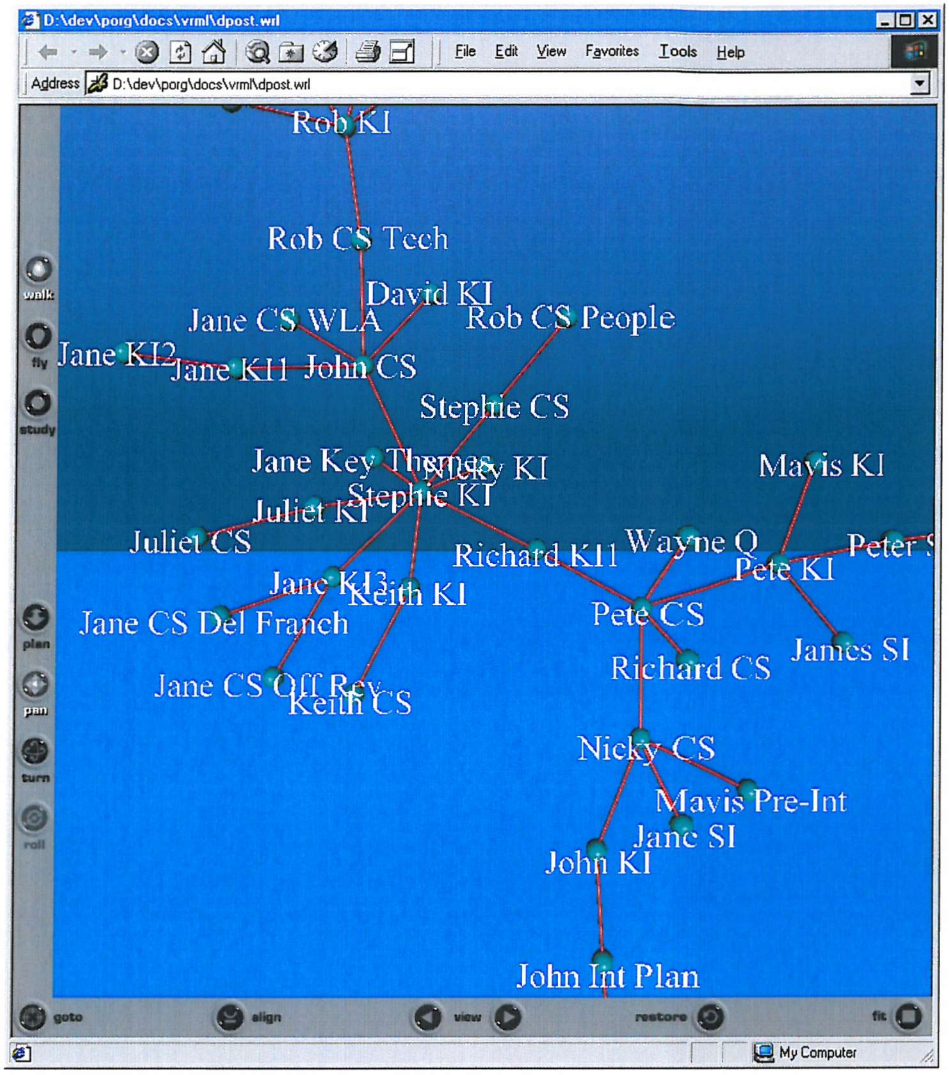


Figure B.2: Example of a Pathfinder Network for Documents Shown in VRML.

It was felt by PORG that the visualisation process had been useful but there were usability issues. The VRML worlds were slow and difficult to navigate using the current computers of the time. An even bigger problem was the company ban on installing software not on an approved list. This included the VRML viewer and the latest version of Internet Explorer that was necessary to use the accompanying Web pages. The result was that the only people who saw the VRML worlds were the researchers on the project. None of the intended recipients could make use of the data. A more important problem was relating the 3D visualisations back to the documents themselves. In the VRML worlds of document-to-document relationships the user could click on a node and be taken to that document. In the keyword-to-keyword visualisations the user could click on a node and be taken to a listing of the 5 most important documents for that keyword. Once in a report there was little aid to actually navigating within the reports or clues on how to make use of the relationships that were being inferred.

At this point I was approached to attempt to build a link service that could make use of the data behind these networks and link into the documents in a more meaningful way.

B.3 Summary of Goals

The project plan was to build a demonstrator system and try to combine the analysis data provided by Chen with an open hypermedia link service. A Web site of the documents would be linked through a set of generic links whose characteristics would convey the relationships between the documents.

The implementation of the system consisted of a Web site of the AKAP documents complemented by an introductory page and contents listing. When the reader opened a document the CA-DLS added popup links to words in the documents.

Not all of Chen's data was available to use or could be utilised in a way that made sense. The obvious starting point for building a system was the table of keywords and the 5 documents of most relevance to the word. Each document had a weighting between 0 and 1 attached to it. The raw data behind the various networks was obtained from Chen as text files.

The proposal was to use this as the basis for a linkbase of generic links, each with 5 destinations and each of those with a weighting. As the user browsed the document collection any of the keywords found in the content of the documents would be linked to the 5 destination documents and there would be some indication of the strength of the relationship given by the weighting.

The second major task of this project was to design and build a linkbase using the complex link form and write the accompanying parser to load it. For an in depth example of the implementation of a complex linkbase using the XLink specification see Section B.6.

B.4 Interface Implementation

Once the original documents had been converted to Web pages the first stage was to develop a method for displaying links with multiple destinations. This was done first so that it would be easier to know what the CA-DLS should generate. The chosen method was to implement a pop-up menu system. The destinations in the menu were ranked and colour coded according to the weighting given by the data. This indicated how close the destination documents were to the current document. A close document in the world indicated a higher priority, this was indicated by a stronger colour of link. The links were also ranked in the menu so that high priority links were at the top of the menu.

An example of the how a popup menu appears in a browser is shown in Figure B.3. In this case the word 'integrate' has been clicked on and a menu has appeared with five links. Four are relatively high priority and one is of very low priority.

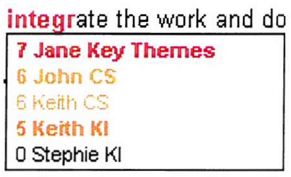


Figure B.3: An Example of the Popup Menus Generated by the Link Resolver.

The menu was implemented using JavaScript written for Internet Explorer. The priority scheme and colouring was written using Cascading Style Sheets (CSS). Figure B.4 shows the HTML placed in a document at the link anchor

```
This link anchor has a priority of 5 <span ID="idPriority5"><a CLASS="clsMenu-
Title" ID = "link8" >click again</a></SPAN>
```

Figure B.4: Sample of HTML Placed in a Document Forming a Prioritised Link Anchor.

When a user clicks on the link anchor JavaScript will load data from the matching DIV data island stored at the bottom of the page. Figure B.5 shows the corresponding DIV data island stored at the bottom of the page.

```
<DIV ID="divMenu8" CLASS="clsMenu">
<A ID="idPriority0" HREF="http://www.iam.ecs.ac.uk">IAM Group Home Page</a>
<A ID="idPriority1" HREF="http://www.ecs.soton.ac.uk/~gvh/">Gareth Hughes</a>
<A ID="idPriority2" HREF="http://muffin.doit.org/">Muffin</a>
...
</DIV>
```

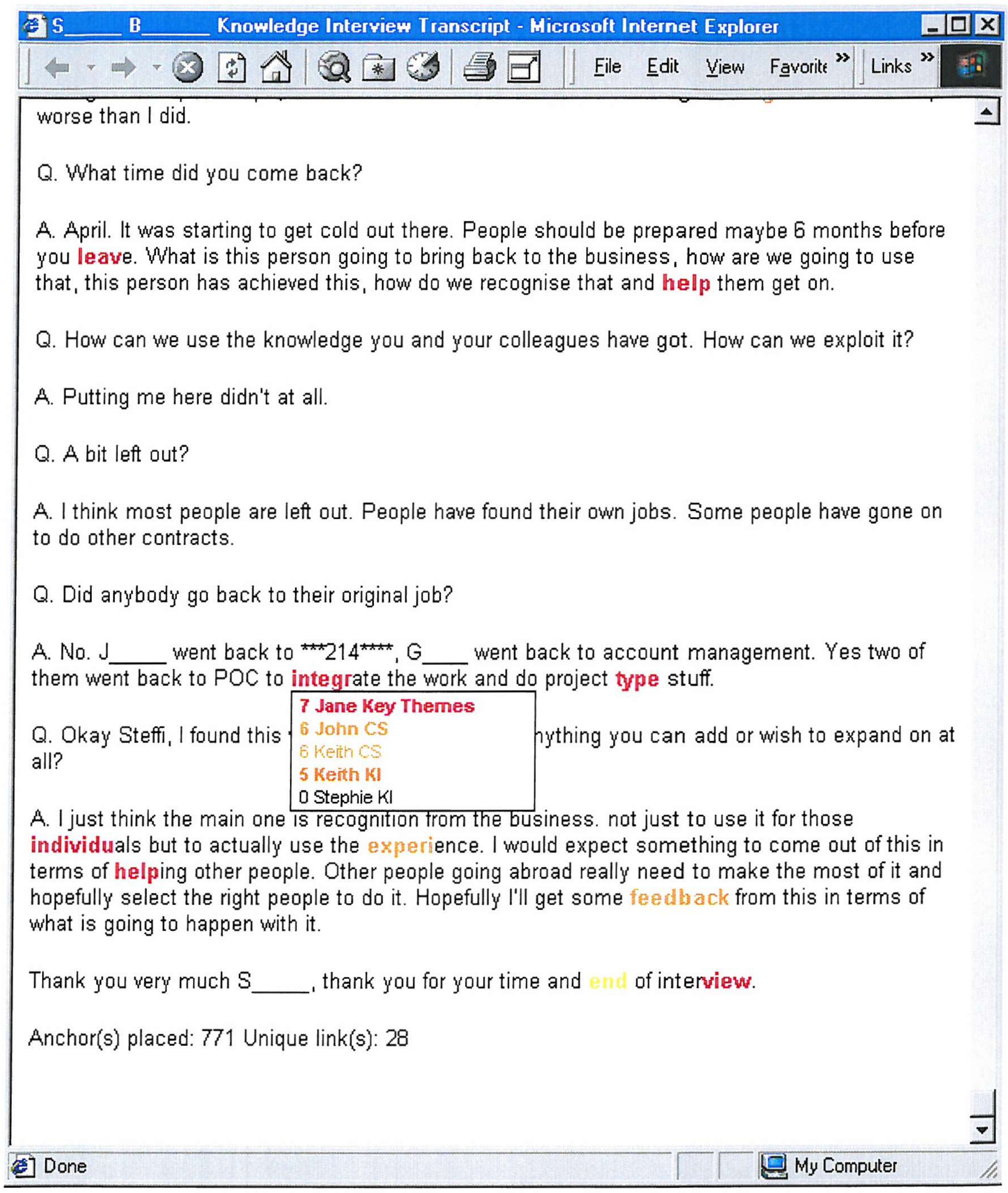
Figure B.5: Sample of an HTML DIV Data Island Depicting the Links to Display in a Popup Menu.

When the user clicks on an anchor the JavaScript loads the data for the menu from the appropriate DIV and displays a menu of links coloured according to the priority.

The colours chosen for the experimental system range from a deep red, denoting highest priority, through oranges to yellow. Ten colours were required and it was always known that finding the best colours to use would be a difficult and controversial task. Hence the system was designed so that colours were only defined in the CSS file and nowhere else. It would be trivial task to alter the file and change the way link priority is denoted. It would also be possible to change the colour of the menu background and borders to aid in designing a better colour scheme. There are many other ways to add formatting using the style sheet, for instance a variety of fonts which could vary in size could be used to convey meanings such as priority or whether the link destination had already been visited. The purpose of the project was primarily to develop a link service architecture and not to be concerned with HCI issues so testing the quality of the colour scheme was not pursued.

When a link anchor was activated a JavaScript function was used to load the appropriate destination data and display it as a popup. The data was stored in a data island stored at the end of the document. Only one occurrence of the data for each link anchor was placed in the document and referred to by a unique identifier. All occurrences of a linked word in the document would use the same data island identifier and hence use the same data island. This gave considerable efficiencies in the document size as well as during the document parsing stage. The parser needed to keep track of what keywords had already been linked and what data islands had been generated. Once an island was generated during the parsing stage there was no further requirement to do so. Data islands would be written out to the document by the *getFooter()* method of the Link Resolver.

Figure B.6 shows an example of a page in which a number of words have been linked. They are the words in colour. The user has activated the link on the word integrate and a menu of destinations has appeared. Some words have been erased for confidentiality reasons.


S_____ B______ Knowledge Interview Transcript - Microsoft Internet Explorer
File Edit View Favorite Links
Done
My Computer

worse than I did.

Q. What time did you come back?

A. April. It was starting to get cold out there. People should be prepared maybe 6 months before you **leave**. What is this person going to bring back to the business, how are we going to use that, this person has achieved this, how do we recognise that and **help** them get on.

Q. How can we use the knowledge you and your colleagues have got. How can we exploit it?

A. Putting me here didn't at all.

Q. A bit left out?

A. I think most people are left out. People have found their own jobs. Some people have gone on to do other contracts.

Q. Did anybody go back to their original job?

A. No. J_____ went back to ***214****, G____ went back to account management. Yes two of them went back to POC to **integr**ate the work and do project **type** stuff.

Q. Okay Steffi, I found this [popup] nything you can add or wish to expand on at all?

7 Jane Key Themes
6 John CS
6 Keith CS
5 Keith KI
0 Stephie KI

A. I just think the main one is recognition from the business. not just to use it for those **individu**als but to actually use the **experi**ence. I would expect something to come out of this in terms of **help**ing other people. Other people going abroad really need to make the most of it and hopefully select the right people to do it. Hopefully I'll get some **feedback** from this in terms of what is going to happen with it.

Thank you very much S_____, thank you for your time and **end** of inter**view**.

Anchor(s) placed: 771 Unique link(s): 28

Figure B.6: A Document Enhanced with Multi-Destination Links.

B.5 Application Overview

Once a method for generating the menus had been decided the system could be developed to produce the required Web pages. The CA-DLS was designed and implemented at this point. The first Link Resolver for the system, called the PorgResolver, was written. A short recap of the overall system functionality is included here.

The user's Web browser is connected to a Web site of the documents via the CA-DLS Web proxy. For each requested document the CA-DLSFilter processes the document looking for keywords from the list it has been given by the PorgResolver and also for other key points in the HTML. Specifically it looks for the <HEAD> and </BODY> elements. Firstly it will find the <HEAD> and at this point invoke the Resolvers *getHeader()* function. This allows the Resolver to add in the HTML needed to include a reference to the JavaScript for the menu system. Once inside the BODY of the document full link processing begins. Each time that a word is found that is on the anchor list the PorgResolver is invoked to resolve that anchor into a link.

When the system starts the PorgResolver loads the following data. The linkbase, a mapping of document file names to an identifier and a matrix of document to document weightings using the same document identifier. When a document is requested the Resolver is informed of the request URL by the system and hence can infer which document the user is viewing. The linkbase data gives the 5 destination documents for the anchor.

For each of these destination documents the Resolver looks up the document-to-document weighting from the currently viewed document to each possible destination. It builds a ranked menu of destination links. This takes the form of a data island to be added to the foot of the Web page. The data is stored by the Resolver until the end of processing when all are added to the bottom of the HTML. The anchor and data are joined by an identifier and if the anchor word is seen again in the document the system will simply reuse the identifier of the previous occurrence. This speeds link processing and minimizes the data added to each page.

B.6 Linkbase Design

The design and building of the linkbase was used as an opportunity to explore emerging standards being produced by the W3C. In the years since Microcosm and the DLS was invented the XLink (DeRose et al., 2000) standard has been created. The standard cites Microcosm as an influence (S. J.

DeRose (Editor), 1999) so it was considered an interesting exercise to use XLink for the linkbase design. The XLink concept of a complex link which can include multiple anchors and destinations in a single link object was an obvious candidate for the multiple destination links required by this work.

A number of standard Java classes for parsing XML and XLink were found and a considerable investment in time was spent understanding and incorporating this code into the system. It was reasoned that the investment would be realised in future applications of the system as the link loading subsystem could be reused with little modification. For each Link Resolver the linkbase design and context model is altered and a new version of the parser and link loader is written to match.

It should be noted that the information about priorities is not stored in the linkbase. Only keywords and their destinations. The datasets produced by Chaomei Chen for document-document distance and keyword-document distance are loaded by the PorgResolver and used to compute the link priorities. The PorgResolver understands which document the user has requested and can look up the priority of each destination document. The PorgResolver contextual model is based around the user's location within the closed document space. The URL of the document the user has requested is the input the system needs to decide how to generate the appropriate links.

Figure B.7 shows one link from the linkbase. It describes an extended link from the word 'integrate' to five destination documents. The priority values in the linkbase are all set to 1 and the real values are stored in another file which was delivered by Chen.

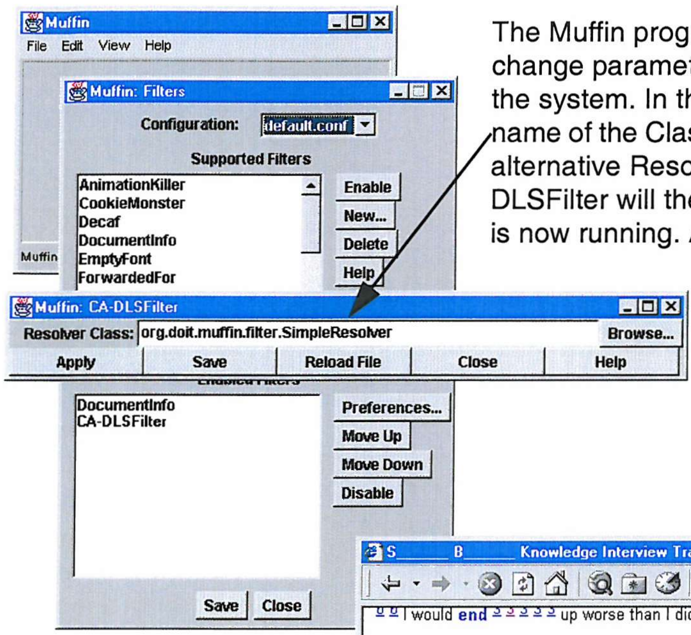
```
<link xml:link= "extended" show="replace" actuate="user" in-line= "false">
<locator show="replace" actuate="user"
  role = "source"
  title = "integrate"
  in-line= "false"
/>
<locator
  role = "destination"
  title = "John CS"
  href = "http://localhost/porg/post-akap/JCCase-Study.htm"
  in-line= "false"
  priority= "1"
/>
<locator
  role = "destination"
  title = "Stephie KI"
  href = "http://localhost/porg/post-akap/SB_KI_transcript.htm"
  in-line= "false"
  priority= "1"
/>
<locator
  role = "destination"
  title = "Jane Key Themes"
  href = "http://localhost/porg/post-akap/JK_AKAP_KEY_THEMES.htm"
  in-line= "false"
  priority= "1"
/>
<locator
  role = "destination"
  title = "Keith CS"
  href = "http://localhost/porg/post-akap/K_F_Case_Study.htm"
  in-line= "false"
  priority= "1"
/>
<locator
  role = "destination"
  title = "Keith KI"
  href = "http://localhost/porg/post-akap/K_F_Knowledge_Interview.htm"
  in-line= "false"
  priority= "1"
/>
</link>
```

Figure B.7: A Linkbase Entry Linking the Word 'integrate' to Five Destinations.

B.7 An Alternative Link Resolver

As part of the process of designing and building the Resolver architecture a second version of the PorgResolver was written that produced a different style of links. This Resolver produced in-line HTML links rather than a menu. This simpler display style ensured that the finished system was compatible with all Web browsers rather than those supporting the JavaScript popup menu code.

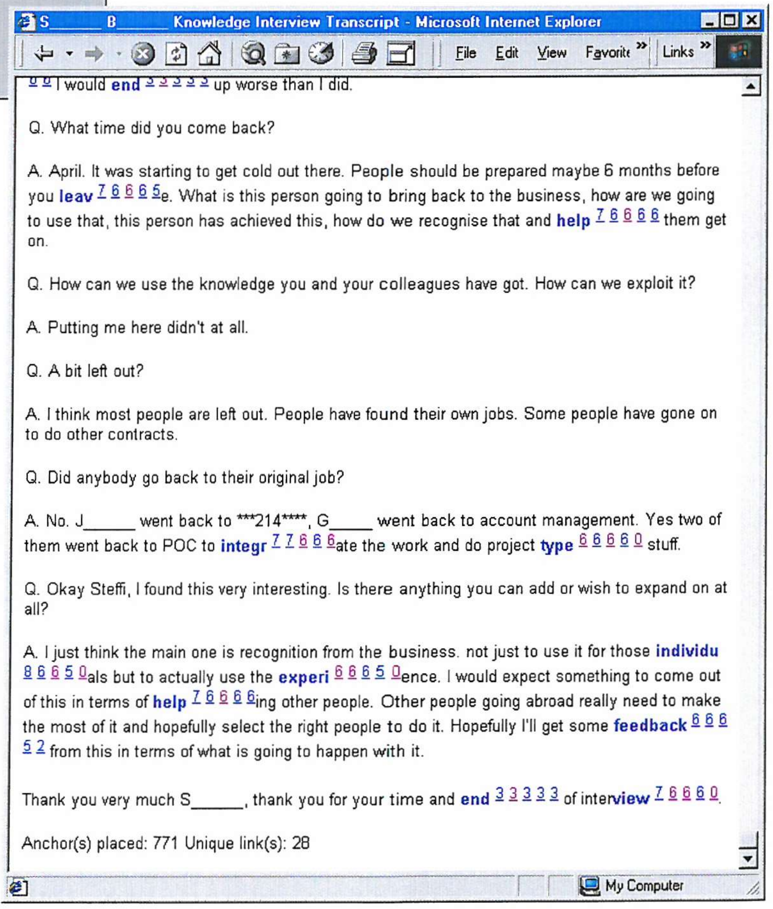
The implementation was written to dynamically allow the user to change the Resolver in use whilst the system was running. A crude technique was implemented to facilitate this. In Figure B.8 below the user enters the name of the class file of the Resolver and the CA-DLSFilter instantiates the new Resolver immediately. The menu based links are replaced with in-line links. It should be noted that the processing and results are the same for both Resolvers as the development was just a proof of concept exercise. The key point of the exercise being to demonstrate the dynamic loading characteristic of the Link Resolver architecture.



Resolver Class: org.doit.muffin.filter.SimpleResolver

The Muffin program provides an interface to change parameters to individual filters running in the system. In this case the user is changing the name of the Class being used as a Resolver. The alternative Resolver is being chosen. The CA-DLSFilter will then reload and the new Resolver is now running. Any pages now viewed will have their links generated by the SimpleResolver.

This is the same document as that shown in Figure B.6 but viewed using the SimpleResolver. The rudimentary way of showing the available links is not meant to be used but just to illustrate the concept of Resolvers.


S_____ B_____ Knowledge Interview Transcript - Microsoft Internet Explorer
File Edit View Favorite Links
My Computer

0 0 I would **end** 3 3 3 3 3 up worse than I did.

Q. What time did you come back?

A. April. It was starting to get cold out there. People should be prepared maybe 6 months before you **leav** 7 6 6 6 5e. What is this person going to bring back to the business, how are we going to use that, this person has achieved this, how do we recognise that and **help** 7 6 6 6 6 them get on.

Q. How can we use the knowledge you and your colleagues have got. How can we exploit it?

A. Putting me here didn't at all.

Q. A bit left out?

A. I think most people are left out. People have found their own jobs. Some people have gone on to do other contracts.

Q. Did anybody go back to their original job?

A. No. J_____ went back to ***214****, G_____ went back to account management. Yes two of them went back to POC to **integr** 7 7 6 6 6ate the work and do project **type** 6 6 6 6 0 stuff.

Q. Okay Steffi, I found this very interesting. Is there anything you can add or wish to expand on at all?

A. I just think the main one is recognition from the business. not just to use it for those **individu** 8 6 6 5 0als but to actually use the **experi** 6 6 6 5 0ence. I would expect something to come out of this in terms of **help** 7 6 6 6 6ing other people. Other people going abroad really need to make the most of it and hopefully select the right people to do it. Hopefully I'll get some **feedback** 6 6 6 5 2 from this in terms of what is going to happen with it.

Thank you very much S_____, thank you for your time and **end** 3 3 3 3 3 of inter**view** 7 6 6 6 0.

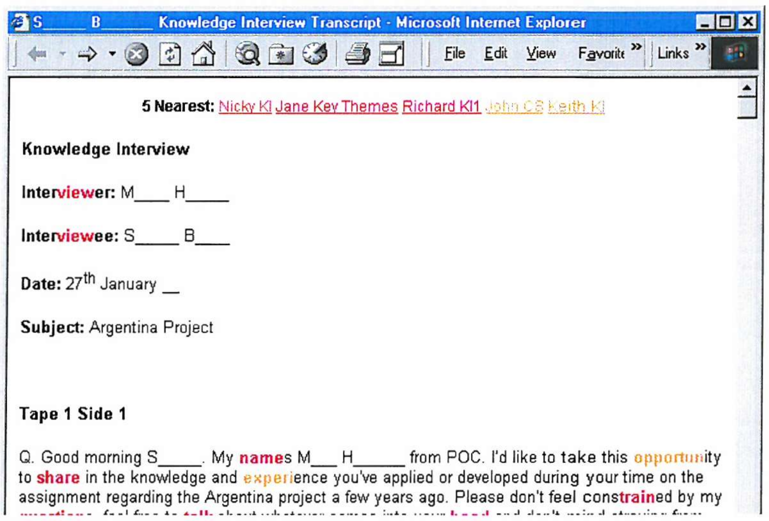
Anchor(s) placed: 771 Unique link(s): 28

Figure B.8: Changing the Link Resolver in use With Muffin.

B.8 5 Nearest Neighbours

Once the system had been built it was a fairly simple matter to adapt the technology to show more of the relationships between the documents. The document-to-document network data provided the means for a simple tool to aid navigation around the document set.

A second Muffin filter was written that added links at the top of the page of each document to the 5 nearest documents in the document space according to the document-document priority data. The same colour scheme and ranking indicator was used helping users evaluate how related the destination document was to one they were reading. This can be seen in action in Figure B.9 below. The 5 links at the top of the document were produced by this filter.


S_____ B______ Knowledge Interview Transcript - Microsoft Internet Explorer
File
Edit
View
Favorite
Links

5 Nearest: Nicky KI Jane Key Themes Richard KI1 John CS Keith KI

Knowledge Interview

Interviewer: M____ H_____

Interviewee: S_____ B____

Date: 27th January __

Subject: Argentina Project

Tape 1 Side 1

Q. Good morning S_____. My **names** M___ H______ from POC. I'd like to take this opportunity to **share** in the knowledge and experience you've applied or developed during your time on the assignment regarding the Argentina project a few years ago. Please don't feel cons**train**ed by my

Figure B.9: The Link Service Adding Five Links to the Nearest Documents in the Space.

B.9 Project Conclusions

The finished system utilised generic links as a way to navigate through the document collection and discover the relationships as found by Chen's analysis work. The data displayed in the 3D VRML worlds was embedded back into the documents allowing users to directly take advantage of the analysis results whilst reading the documents.

When the delivered VRML worlds and data had been analysed it had proved difficult to derive a great deal of worth from them alone as they did not relate easily back to the original documents. Now that the source documents were enhanced with the data it became an easier task to examine them and question how well the knowledge capture process had worked. The original reason for doing all of the work was to capture business knowledge from the people who went to Argentina on the contract. Could the important knowledge be found within the documents and delivered to the right people within the organisation? Informal trials of the system with members of PORG quickly showed that the answer was no. The reason lay back with the knowledge capture process and the resulting documents.

Many of the documents were extremely long. In the 42 documents used in the work there were 284384 words amounting to 680 pages. For a reader to find the important pieces of knowledge from such an enormous amount of material was an extremely difficult process. The bigger problem for the analysis projects that followed was that they relied on using keywords to describe the whole document. This would be hard enough for someone who was a domain expert and would know what keywords or phrases to look for. However the keywords used were chosen by Chaomei Chen who, not being a domain expert, did not choose the best set. Needless to say the keywords could be chosen again by an expert and Chaomei could run all of his analysis work again. The underlying problem is the assumption that a long document can be described by five keywords. The major limiting factor is still the length of the documents.

A possibility to consider would be to decompose the documents into a vast number of smaller documents and analyse those. This would have the advantage that a keyword would be able to more accurately describe the smaller document. Whilst visualisation of a very large number of documents would pose Chen and users far greater problems it would be to the advantage of the link service. The system would fall down because it would be difficult to decide how to partition documents automatically. These were conclusions that the AKAP team had also come to. There were lessons to be learnt for the team who had undertaken the knowledge capture process in the first place.

The decision to use open software in the form of Muffin and various XLink parsing libraries brought advantages and disadvantages. The main reason for using open software was to speed development of the code and to concentrate on the research issues, not the implementation of basic system infrastructure. This worked initially allowing the CA-DLS to be running in a short time. However there were a number of serious bugs in the code that took a lot of time to find and solve. Thus the time taken to code the system was approximately the same as if it has been written from scratch, but the majority of the time had been spent on the areas of interest.

At the end of the project I suggested that the best way to proceed would be to manually split the documents up into much smaller fragments. These should then be analysed by a human to decide what parts of each were of value and to assign keywords. What I did not know at that time that this had already happened at PORG. The results of that analysis work form the input to the second PORG project described in the next appendix.

Appendix C: Implementation of Chapter Six, Part Two.

This appendix describes a second project carried out with the Post Office Research Group (PORG) which followed on from the work in Appendix B. This project used the results of another analysis effort carried out by a member of PORG. The same document set was analysed by hand in a laborious attempt to find the best information in the documents. This involved physically reading all of the documents, finding the most important excerpts and forming an overall picture of the content. A document was produced, the 'Learning Summary' report, that contained an analysis of the documents and an in-depth summary of the knowledge found. The major contribution was a number of diagrams they called Knowledge Maps (KM), accompanied by references and indices. See Figure C.2 for an example. A KM is a hierarchical tree diagram of areas and sub-areas of knowledge covered in the document set. One KM, covering technical areas, is backed up by full references to the actual locations of relevant content on that topic. This analysis is of high quality and made good use of the documents.

As before the figures in this appendix have been altered as necessary to protect privacy and confidentiality.

C.1 Project Summary

At the time of completion of the first project I had no idea that this analysis work had been done or that the 'Learning Summary' existed. The brief of the project was open ended, I was given the report and KM maps with no request from the external partner for any particular deliverable. My starting point was the future work recommendations I had given them at the end of the previous project. The major recommendation was to break the documents up and work on a paragraph or page level analysis. This tallied with the work done for the 'Learning Summary' so it was natural to begin working on a way to implement some way of achieving this.

There were no pre-set requirements and no idea of what the outcome of the work should be. Therefore the project started with an in-depth study of the 'Learning Summary' and a survey of technologies and standards that could be used in an implementation. There was a desire that the project should be an opportunity to showcase and evaluate a number of standards being published by the W3C. There was not the same need for any software produced to be backwards compatible. The source materials were studied and a great deal of research was done into various programming technologies until a plan evolved and a schedule of deliverables was written.

A key factor in the project planning process was that a great deal of work had gone into the analysis work but it was just shown as diagrams in the paper report such as Figure C.2. The diagrams were clear, well designed and formed a good overall understanding of the knowledge found. However the results were essentially 'locked' within this set of Microsoft Word diagrams and the data they contained was not reusable. The prime motivation was to demonstrate that technologies such as open hypermedia and open standards could bring such work to life as well as enable knowledge reuse. On the implementation front the remit went beyond just extending the architecture of the CA-DLS.

I set an overall project goal to reproduce the 'Learning Summary' report as a living, dynamic Web site. Each part of the report was recreated using a variety of techniques to ensure that the knowledge could be reused and displayed as needed. The Knowledge Maps and references were re-written using XML and transformed to Web pages using XSLT transforms. This allowed for superior display techniques to allow users to view the KM's as well as allowing them to link into the documents or other data as required. The goal was to show that open data formats could present a superior alternative to a paper based report.

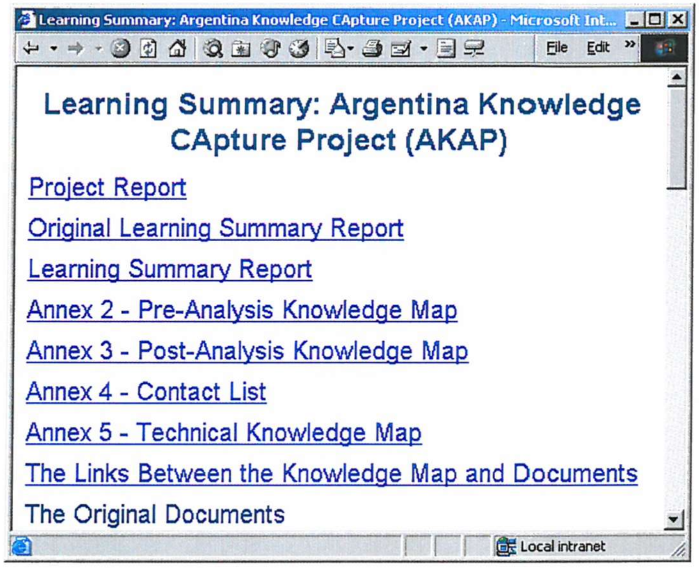
The 'Learning Summary' report was designed to accompany and reference the original AKAP documents described in the previous project. In order to use them with this project they were converted to XML, a difficult and time consuming process in itself.

During the project a second major Link Resolver was developed for processing a linkbase and activating XPointer links. The Resolver was utilised both using a standalone program and the existing CA-DLS. The standalone program was used to batch process the linkbase due to its size and the complexity of the XML processing being done. Due to the Link Resolver design the system also worked when activated through the CA-DLS.

C.2 Project Implementation

Each of the sections of the 'Learning Summary' were converted or adapted to be accessible from a Web browser. The main body of the report was a simple text report but following that were all of the KM diagrams and data to support them. Each was represented as XML files and accompanied by XSL files to render them viewable. This section lists the KM's and demonstrates the transformation process.

The site contains the following sections which map to the parts of the original report.


Learning Summary: Argentina Knowledge CApture Project (AKAP) - Microsoft Int...
File Edit
Local intranet

Learning Summary: Argentina Knowledge CApture Project (AKAP)

Project Report
Original Learning Summary Report
Learning Summary Report
Annex 2 - Pre-Analysis Knowledge Map
Annex 3 - Post-Analysis Knowledge Map
Annex 4 - Contact List
Annex 5 - Technical Knowledge Map
The Links Between the Knowledge Map and Documents
The Original Documents

Figure C.1: The Contents Page of the Live Version of the Learning Summary Report.

Section	Explanation
Learning Summary Report	The main body of text from the report, reproduced as HTML.
Annex 2 - Pre-Analysis Knowledge Map	These knowledge maps are written in XML and reproduce the diagrams produced in the report.
Annex 3 - Post-Analysis Knowledge Map	

Table C.1: The Sections of the Learning Summary Web Site.

Section	Explanation
Annex 4 - Contact List	The contact list has been merged with all other acronyms used in the report to form a reusable XML resource. The Knowledge maps dynamically use the data in the contact list to expand acronyms and initials to their full values.
Annex 5 - Technical Knowledge Map	A KM depicting the technical subjects found within the document set.
The Links Between the Knowledge Map and Documents	This is the core of the project and is described below.
The Original Documents	The XML versions of the original documents that this analysis is based on.

Table C.1: The Sections of the Learning Summary Web Site.

C.3 Document Conversion

The first task was to convert the original AKAP documents used in the previous project from Microsoft Word format to XML. Unfortunately the work was hampered as in the previous project by the poor state of the supplied documents. There was no consistency in the writing style and no use of Word styles used in any document. Subsequently the work took much longer than planned.

The document set of Microsoft Word files was imported into Adobe Framemaker 6 and headings added to the text where economic to do so. A package called Webworks Publisher included with Framemaker 6 was used to export each document to XML. The package creates a complex XML document and makes use of a Cascading Style Sheet file to achieve a final document that can be displayed on any version 5 Web browser. A key property of the generated XML is that each paragraph is given a unique identifier. These XML documents were used as the basis for the project.

C.4 The Analysis Knowledge Maps

In the original 'Learning Summary' the various Knowledge Maps were drawn using Microsoft Word. An example is shown in Figure C.2. In the report there are 3 KM's. Each was replicated as an XML document. Figure C.3, shows a fragment of XML from the same Knowledge Map.

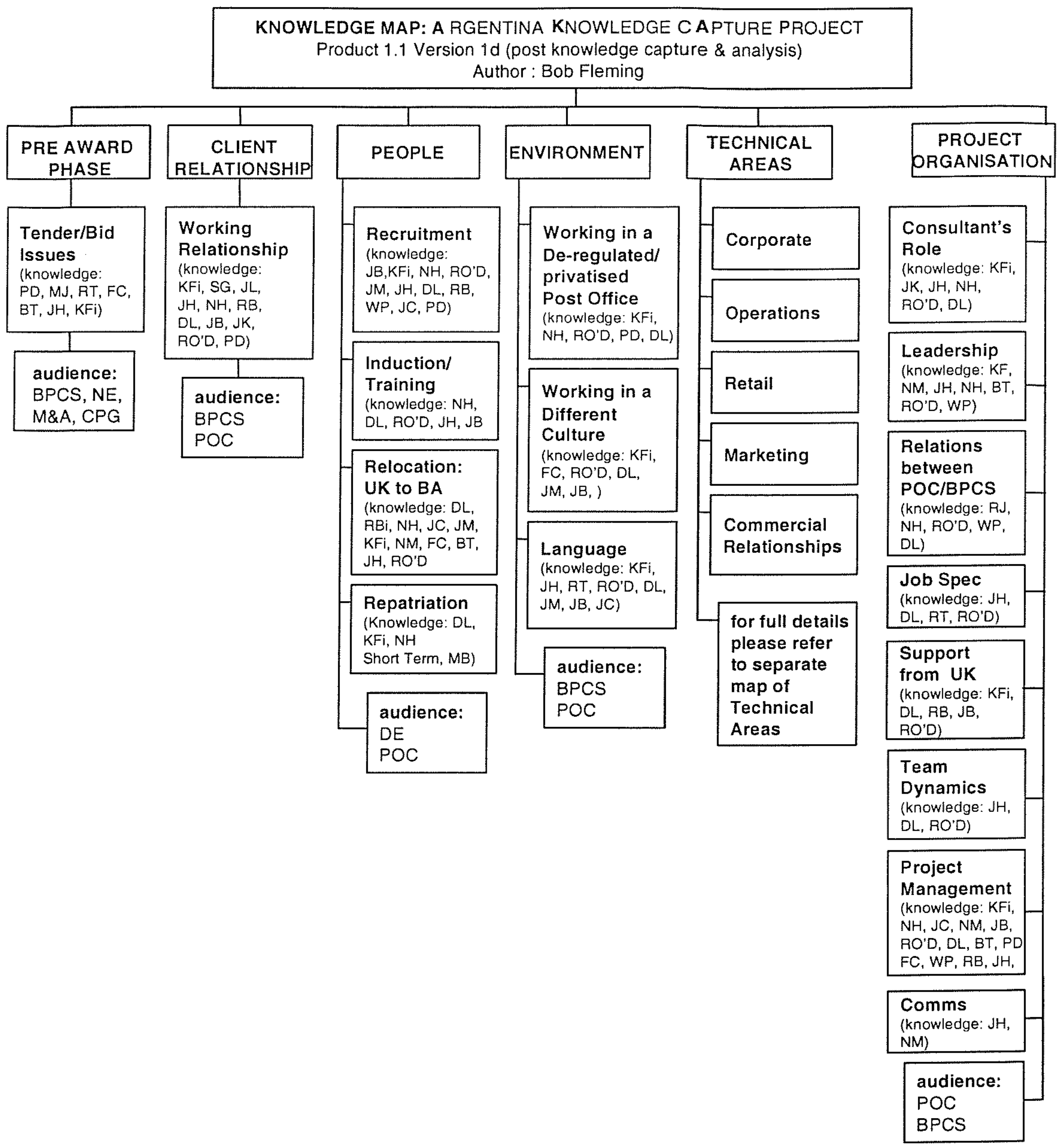

KNOWLEDGE MAP: A RGENTINA KNOWLEDGE C APTURE PROJECT
Product 1.1 Version 1d (post knowledge capture & analysis)
Author : Bob Fleming
PRE AWARD PHASE
CLIENT RELATIONSHIP
PEOPLE
ENVIRONMENT
TECHNICAL AREAS
PROJECT ORGANISATION
Tender/Bid Issues (knowledge: PD, MJ, RT, FC, BT, JH, KFi)
audience: BPCS, NE, M&A, CPG
Working Relationship (knowledge: KFi, SG, JL, JH, NH, RB, DL, JB, JK, RO'D, PD)
audience: BPCS POC
Recruitment (knowledge: JB,KFi, NH, RO'D, JM, JH, DL, RB, WP, JC, PD)
Induction/ Training (knowledge: NH, DL, RO'D, JH, JB
Relocation: UK to BA (knowledge: DL, RBi, NH, JC, JM, KFi, NM, FC, BT, JH, RO'D
Repatriation (Knowledge: DL, KFi, NH Short Term, MB)
audience: DE POC
Working in a De-regulated/ privatised Post Office (knowledge: KFi, NH, RO'D, PD, DL)
Working in a Different Culture (knowledge: KFi, FC, RO'D, DL, JM, JB,)
Language (knowledge: KFi, JH, RT, RO'D, DL, JM, JB, JC)
audience: BPCS POC
Corporate
Operations
Retail
Marketing
Commercial Relationships
for full details please refer to separate map of Technical Areas
Consultant's Role (knowledge: KFi, JK, JH, NH, RO'D, DL)
Leadership (knowledge: KF, NM, JH, NH, BT, RO'D, WP)
Relations between POC/BPCS (knowledge: RJ, NH, RO'D, WP, DL)
Job Spec (knowledge: JH, DL, RT, RO'D)
Support from UK (knowledge: KFi, DL, RB, JB, RO'D)
Team Dynamics (knowledge: JH, DL, RO'D)
Project Management (knowledge: KFi, NH, JC, NM, JB, RO'D, DL, BT, PD FC, WP, RB, JH,
Comms (knowledge: JH, NM)
audience: POC BPCS

Figure C.2: A Sample Knowledge Map (KM) from the Original Learning Summary Report.

This XML fragment is from the Post Analysis Knowledge Map.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="analysis.xsl"?>

<KM
title="Post Analysis Knowledge Map (Annex 3)"
version="Product 1.1 Version 1d (post knowledge capture and analysis)"
project="Argentina Knowledge Capture Project"
author="Bob Fleming"
>
<Area name="Pre Award Phase" id="1">
<Audience>BPCS</Audience>
<Audience>NE</Audience>
<Audience>M&amp;A</Audience>
<Audience>CPG</Audience>
<Sub-Area name="Tender/Bid Issues" id="1.1">
  <Knowledge id="1.1.1">PD</Knowledge>
  <Knowledge id="1.1.2">MJ</Knowledge>
```

Figure C.3: A Fragment of XML from the Technical Analysis KM.

In order to view the raw XML versions of the Knowledge Maps an XSL stylesheet had to be designed which would display the Knowledge Map in a similar fashion to the original diagrams. The highlights of the XSL file to do that is shown in Figure C.4. Due to the nature of XSL it is possible to achieve the same effect by writing this file in a number of ways and many improvements might be possible.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<HTML>
  <BODY>
  <P class="Title"><xsl:apply-templates/> </P>

  <TABLE WIDTH="100%" BORDER="0" >
  <TR VALIGN="TOP">
  <xsl:for-each select="KM/Area">
     <TD><P class="CellHeading"><xsl:value-of select="@name"/> </P></TD>
  </xsl:for-each>
  </TR>
  <TR VALIGN="TOP">
  <xsl:for-each select="KM/Area">
     <TD>
     <xsl:for-each select="Sub-Area|Audience">
     <xsl:variable name="foo"><xsl:value-of select="@name" /></xsl:variable>
        <A>
        <xsl:attribute name="HREF">
           ../output/<xsl:value-of select="@name"/>.xml
           </xsl:attribute><xsl:value-of select="@name"/>
        </A>
        <BR />
        <xsl:for-each select="Issue|Knowledge">
           ...
           <A>
           <xsl:attribute name="HREF">
              ../output/<xsl:value-of select="$foo"/>
                 -<xsl:value-of select="."/>.xml
           </xsl:attribute><xsl:value-of select="."/>
           </A>
           <BR />
        </xsl:for-each>
     </xsl:for-each>
     </TD>
  </xsl:for-each>
  </TR>

  </TABLE>
  </BODY>
  </HTML>
</xsl:template>

<xsl:template match="KM">
  <xsl:value-of select="@title"/>
</xsl:template>

</xsl:stylesheet>
```

Figure C.4: The XSL File Needed to Display an XML Knowledge Map on a Web browser.

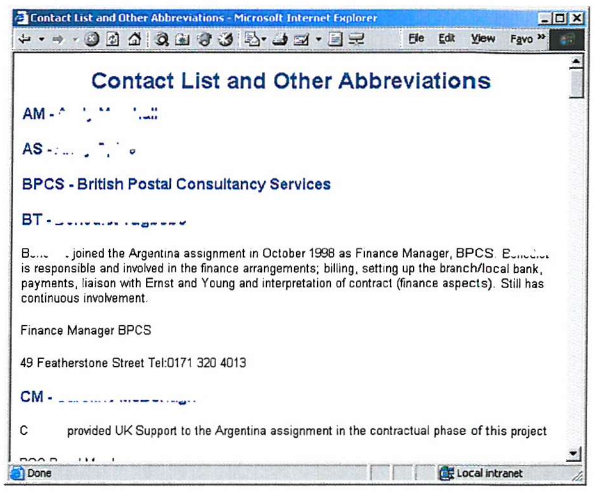
When this XSL file is used to transform a Knowledge Map XML file the result is a Web page. This Web page contains all of the names of the people in the project so is not shown in this thesis for confidentiality reasons.

C.5 The Contact List Knowledge Map

The Contact List Knowledge Map is a list of people and acronyms used throughout the report. It matches the abbreviations used in the Pre and Post Analysis Knowledge Maps to the full names of the person or department. This reusable data source can then be either read by humans, via an appropriate XSL stylesheet, or used by other applications as a data source. For instance the previous section showed one of the KMs in which the initials of users and other acronyms had been expanded out. This is possible because the XSL language has a powerful ability to use data from a second XML source whilst processing a document. Figure C.5 shows a single entry for a person from the XML version of the Contact List and Figure C.6 shows the Contact List when viewed with a Web browser.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="contact_list.xsl" type="text/xsl"?>
<Contacts title="Contact List and Other Abbreviations">
<Person abbreviation="AM">
   <Name>Ally McBeal</Name>
   <Argentina_Role></Argentina_Role>
   <Current_Role></Current_Role>
   <Contact_Details></Contact_Details>
</Person>
```

Figure C.5: A Fragment of the XML of the Contact List KM.


Contact List and Other Abbreviations - Microsoft Internet Explorer
File
Edit
View
Favo
Done
Local intranet

Contact List and Other Abbreviations

AM - [redacted]

AS - [redacted]

BPCS - British Postal Consultancy Services

BT - [redacted]

B[redacted] joined the Argentina assignment in October 1998 as Finance Manager, BPCS. B[redacted] is responsible and involved in the finance arrangements; billing, setting up the branch/local bank, payments, liaison with Ernst and Young and interpretation of contract (finance aspects). Still has continuous involvement.

Finance Manager BPCS

49 Featherstone Street Tel:0171 320 4013

CM - [redacted]

C[redacted] provided UK Support to the Argentina assignment in the contractual phase of this project

Figure C.6: The Contact List KM Viewed Through a Web browser

This is a prime example of how open standards encourage information reuse. In an ideal world the Contact List should be generated automatically from a source such as a personnel database and not be re-written by the report author.

C.6 Linkbase Design

The major implementation task in this project was to design and build a tool to make use of the data stored in the Knowledge Maps and link it back to the raw documents. Before that could be done a great deal of design work was required to learn how to build the linkbase and other data sources. The analysis work was represented as a complex linkbase linking concepts in a Knowledge Map to the relevant paragraphs in the source documents. The finished tool would be able to follow the links in the linkbase and find the destination paragraphs. The tool would use these paragraphs to build new documents on topics described by the Knowledge Maps. The final system can build a new document on a topic or subtopic in the Knowledge Map. For instance a document can be generated containing all of the most important paragraphs in the documents on the subject of purchasing agreements. This tool was built using the CA-DLS and Link Resolver architecture.

One of the core pieces of analysis contained in the 'Learning Summary' was a Knowledge Map on technical areas covered in the document set. This was followed by 9 pages of listings detailing where in the document set the knowledge was contained. The data lists pages of the documents which are considered to be worth reading for a particular topic or sub-topic. Figure C.7 is an example of such data.

OPERATIONS	KEYWORDS	DOCUMENT	NAME	PAGE
NEW PROCESSING CENTRE	New Mail Centre	Case Study [AKAP-RO'D-case4-technicalareas.doc]	R__ O'D______	All
	CTP	Knowledge Transfer Report [AKAP-RO'D-KnowledgeReport.doc]	R__ O'D_____	All
	Benchmarking	Case study	P___ D_____	2
Lean Team Management	MNI	AKAP Knowledge Questionnaire	F____ C______	3
Automation	Sorting machinery	Case Study [AKAP-RO'D-case4-technicalareas.doc]	R__ O'D______	2, 3, 4
		Knowledge Transfer Report [AKAP-RO'D-KnowledgeReport.doc]	R__ O'D______	6-10

Figure C.7: A Sample of Source Data for Building the Linkbase.

The task was to design a representation of this information as a set of links. The links would go from a point in a knowledge map to a paragraph or paragraph range in one of the source documents. The design process proceeded in three stages.

The first stage was to convert paper-based page numbers to XML file-based paragraph numbers for each link. These formed the destination ranges for each link.

The second stage was to define XPath statements that represented these paragraph ranges as the end point of each link. The intention being that the destination of a link would be an XPath statement that a suitable engine could apply to the appropriate document to extract the paragraphs in question.

The third stage was to design a linkbase to represent the complex link design. The source of the link being a point in a KM and the destinations being paragraphs defined by XPath destination statements. The complex form of links as defined in the XLink specification was used. The overall goal was to write a system that could follow links in a Knowledge Map to the specific paragraphs located within the AKAP documents and deliver those paragraphs to the user or another system.

A way of representing this data in an XML linkbase was investigated and then designed. The data in the first two columns is the data in the XML Knowledge Map. The destinations of each link are to the newly created XML versions of the documents. The original links point to pages in the text. These no longer exist in the XML so for each link a paragraph or paragraph set was specified. This analysis was carried out by hand resulting in the data in Figure C.8.

For each anchor an id has been added and for each page or page range a paragraph number or paragraph range has been created. For certain entries there were multiple ranges which would be represented as multiple link destinations. Each area and sub-area has an id as well.

OPERATIONS	KEYWORDS	DOCUMENT	NAME	PAGE
NEW PROCESSING CENTRE **2.1**	New Mail Centre	Case Study [AKAP-RO'D-case4-technica-lareas.doc]	R__ O'D______**55**	All
	CTP	Knowledge Transfer Report [AKAP-RO'D-KnowledgeReport.doc]	R__ O'D______ **56**	All
	Benchmarking	Case study	P__ D______ **57**	2 **246-249**
Lean Team Management 2.1.1	MNI	AKAP Knowledge Questionnaire	F___ C______ **58**	3 **330**
Automation	Sorting machinery	Case Study [AKAP-RO'D-case4-technica-lareas.doc]	R__ O'D______ **59**	2, 3, 4 **231-251**
		Knowledge Transfer Report [AKAP-RO'D-KnowledgeReport.doc]	R__ O'D______ **60**	6-10 **470-561**

Figure C.8: Sample of Source Data for Link with XML Paragraph Numbers Added.

The linkbase designed for this project consists of separate elements describing a link in three parts. There are a set of start points and a set of end points which are joined together by actual links. In the linkbase these are referred to as 'keyword', 'paragraph' and 'go'.

A 'keyword' represents a point on the Knowledge Map using an XPath statement.

Each 'paragraph' entry represents a location in a document as defined by the analysis. The actual destination paragraphs to use are specified using an XPath statement. When this XPath transform is applied to the original document the result is just the specified paragraphs as XML.

A 'go' is an actual link. It states that there is a connection between a point on the Knowledge Map and a paragraph set in one of the original documents. When the linkbase is loaded it represents each 'go' as a link object to be used and manipulated as required.

This linkbase design follows the XLink standard. The standard does not specify that a link should have a particular set of components or be laid out as below. It merely specifies the details of the language to use. The linkbase still has to be designed and the application written to use it.

Below is a sample of the actual linkbase showing the variety of elements found.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="linkbase.xsl" type="text/xsl"?>
<!--Represents all the knowledge in the knowledge map and links-->
<!DOCTYPE link SYSTEM "km.dtd">

<linkbase title="Porg 2 Linkbase">
<paragraph
href="http://localhost/porg2/xml/RD-CaseStudy4-TechnicalAreas.xml"
id="55"
title="para_title"
/>
<paragraph
href="http://localhost/porg2/xml/PD_case-study.xml#//wp:Document[1]/wp:Con-
tent//*[(number(@wp:id)&gt;=999246 and number(@wp:id)&lt;=999249)]"
paragraphs="999246-999249"
id="57"
title="para_title"
/>
<keyword
href="http://localhost/porg2/km/tech_km.xml#///*[@id='2.1']"
id="2.1"
title="New Processing Centre"
/>
<go
from="2.1"
to="55"
title="New mail centre"
arcrole="arcrole"
/>
...
...
</linkbase>
```

Figure C.9: A Fragment of the XML for the Main Linkbase of this Application.

As explained previously the destination URLs were supposed to conform to the XPointer specification but no software supported this. Therefore the value of a 'href' comprised a URL followed by a '#' and an XPath statement to be processed by an XSL engine.

```
href="http://localhost/porg2/xml/PD_case-study.xml#//wp:Document[1]/wp:Con-
tent//*[(number(@wp:id)&gt;=999246 and number(@wp:id)&lt;=999249)]"
```

Figure C.10: A Sample Link Destination from the Main Linkbase.

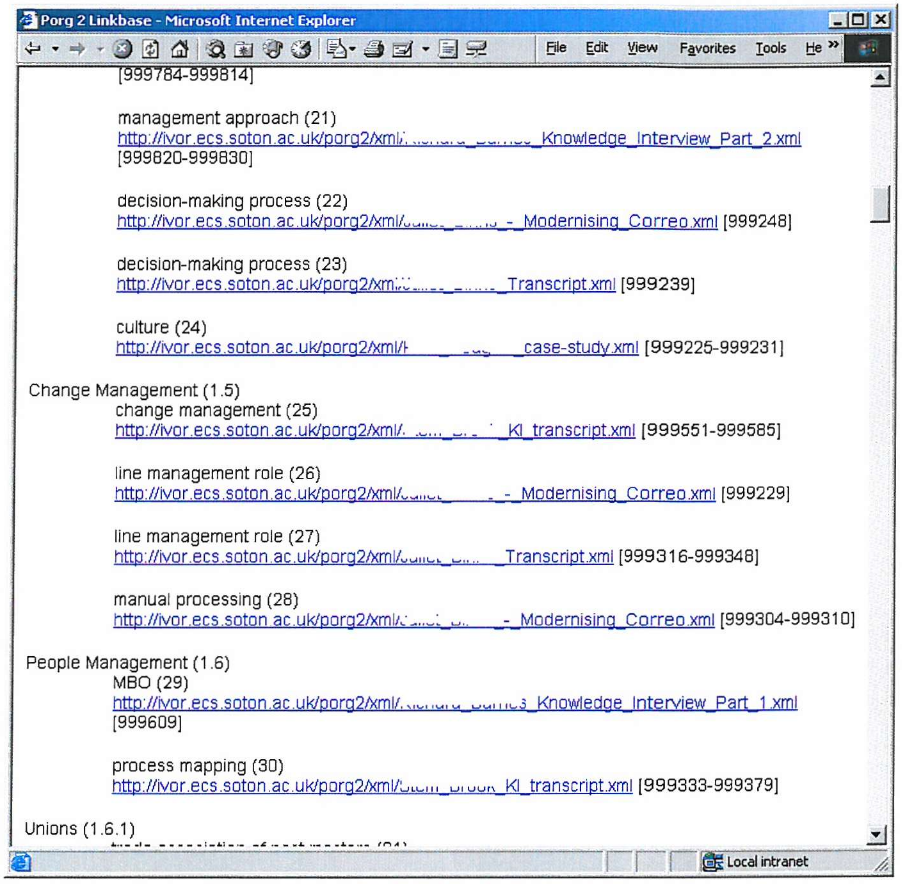
The XPath statement in this link is shown below in Figure C.11.

```
//wp:Document[1]/wp:Content//*[(number(@wp:id)&gt;=999246 and
number(@wp:id)&lt;=999249)]
```

Figure C.11: The XPath Statement Within Each Link Destination.

This statement can be translated as follows. 'Find all elements in the wp:Document/wp:Content branch where the wp:id value of the element is between 999246 and 999249.' Each paragraph has been given an id (wp:id) by Webworks Publisher. This statement takes a significant amount of processing and is further slowed by the considerable size of many of the XML documents. Therefore the processing of the whole linkbase takes some number of minutes.

In the linkbase there are 200 'paragraph' entries, 65 'keyword' entries and 201 'go' entries, or actual links. The linkbase design assigns one 'keyword' entry for each section or subsection of the Knowledge Map. There is one destination entry for each destination written about by the 'Learning Summary' author. The 'go' entries bind from a source 'keyword' to a 'paragraph' destination. In many cases a single 'keyword' will have multiple destination 'paragraphs' and the model suits this well. All of the links were created by manually writing the XLink file. This linkbase can also be viewed as a Web page after transformation by a specially written XSL file. Figure C.12 shows the transformed linkbase.


Porg 2 Linkbase - Microsoft Internet Explorer
File Edit View Favorites Tools He
Local intranet

[999784-999814]

management approach (21)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_Knowledge_Interview_Part_2.xml
[999820-999830]

decision-making process (22)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_-_Modernising_Correo.xml [999248]

decision-making process (23)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_Transcript.xml [999239]

culture (24)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_case-study.xml [999225-999231]

Change Management (1.5)
change management (25)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_KI_transcript.xml [999551-999585]

line management role (26)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_-_Modernising_Correo.xml [999229]

line management role (27)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_Transcript.xml [999316-999348]

manual processing (28)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_-_Modernising_Correo.xml [999304-999310]

People Management (1.6)
MBO (29)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_Knowledge_Interview_Part_1.xml
[999609]

process mapping (30)
http://ivor.ecs.soton.ac.uk/porg2/xml/[illegible]_KI_transcript.xml [999333-999379]

Unions (1.6.1)
trade association of port masters (31)

Figure C.12: The Main Linkbase as Viewed in a Web browser.

The separation of the different components of a link makes reuse of the data and linkbase parser code much easier. When an XML parser loads the file it will build a collection of each type of element and then bind them together to form link objects as required. Object oriented code design makes it possible to add a new element type or alter the components of an existing one. It also makes it an easier task to add other types of data into a link object. For instance a link object might also contain a security rating. This linkbase format and matching code has been the base for all the later Link Resolver examples.

C.7 The Implementation of a Knowledge Map Reader (KMReader)

The major project goal was to build a program that could read the Technical Analysis Knowledge Map and use the matching linkbase to find the specified paragraphs from the original documents. The paragraphs would be compiled into new documents representing the definitive knowledge on a given topic.

C.8 The KMReader

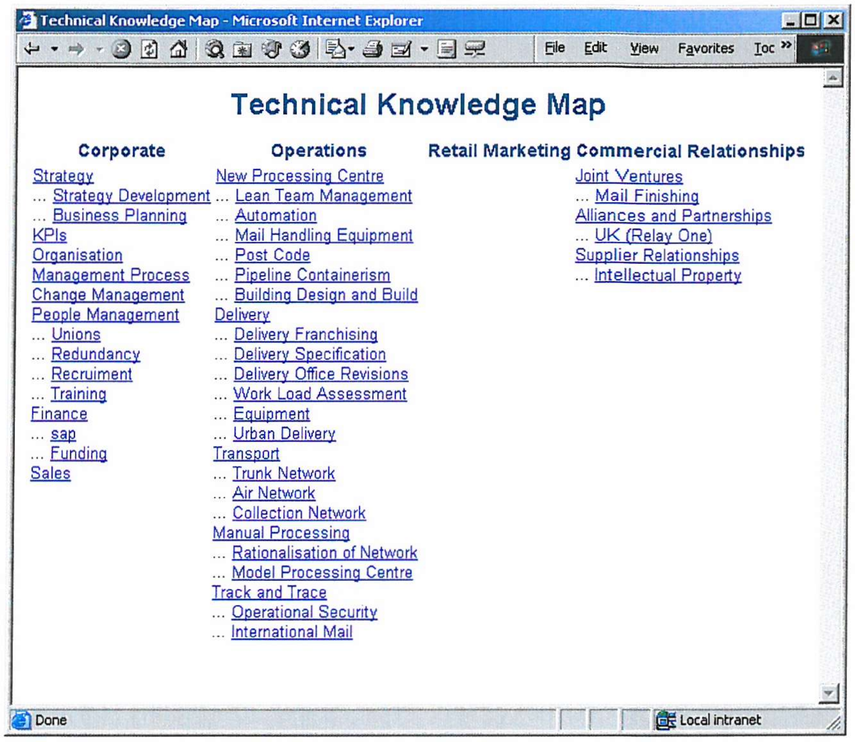
The KMReader is a standalone Java application that processes the linkbase and generates new documents from the Knowledge Map and linkbase. It is a harness for the XMLFragmentResolver and uses the same function calls to the Resolver as the CA-DLS.

A fragment of the XML of the Technical Knowledge Map is shown in Figure C.13. The rendered version is shown in Figure C.14.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="km.xsl"?>

<KM
title="Technical Knowledge Map"
project="Argentina Knowledge Capture Project"
author="Ron Manager"
>
<Area name="Corporate" id="1">
<Sub-Area name="Strategy" id="1.1">
   <Issue id="1.1.1">Strategy Development</Issue>
   <Issue id="1.1.2">Business Planning</Issue>
</Sub-Area>
<Sub-Area name="KPIs" id="1.2">
</Sub-Area>
...
```

Figure C.13: A Fragment of the Technical Knowledge Map as XML.


Technical Knowledge Map - Microsoft Internet Explorer

Technical Knowledge Map

Corporate

Strategy
... Strategy Development
... Business Planning
KPIs
Organisation
Management Process
Change Management
People Management
... Unions
... Redundancy
... Recruiment
... Training
Finance
... sap
... Funding
Sales

Operations

New Processing Centre
... Lean Team Management
... Automation
... Mail Handling Equipment
... Post Code
... Pipeline Containerism
... Building Design and Build
Delivery
... Delivery Franchising
... Delivery Specification
... Delivery Office Revisions
... Work Load Assessment
... Equipment
... Urban Delivery
Transport
... Trunk Network
... Air Network
... Collection Network
Manual Processing
... Rationalisation of Network
... Model Processing Centre
Track and Trace
... Operational Security
... International Mail

Retail Marketing

Commercial Relationships

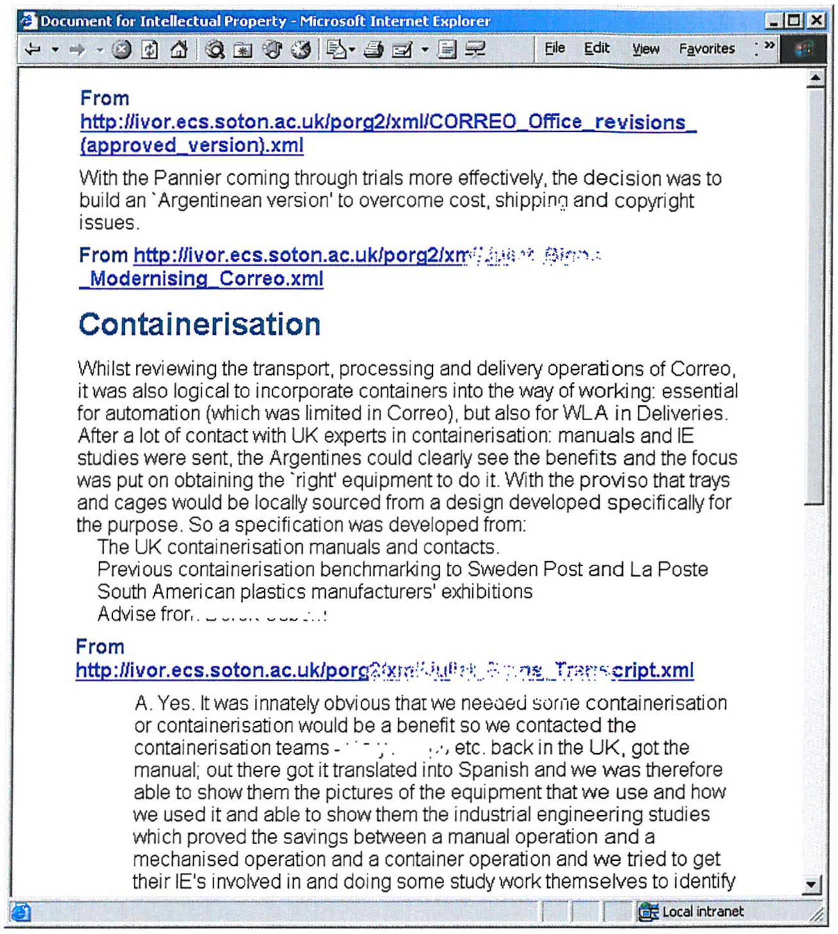
Joint Ventures
... Mail Finishing
Alliances and Partnerships
... UK (Relay One)
Supplier Relationships
... Intellectual Property

Figure C.14: The Technical KM Viewed in a Web browser.

The Knowledge Map in Figure C.14 is similar to the other 2 Knowledge Maps but the XSL stylesheet is more powerful and makes each Area and Sub-Area in the map a link to a document of the same name.

The KMReader loads the Knowledge Map and generates a new document of the same name for each Area and Sub-Area. For instance a file called STRATEGY.XML is created containing all of the paragraphs referenced in its sub-areas of Strategy Development and Business Planning. The program also creates individual files for each of these Sub-Areas as well, 'STRATEGY DEVELOPMENT.XML' and 'BUSINESS PLANNING.XML'.

The documents are formed using a template XML file. The KMReader uses this as a starting point and inserts each result of the XSL transform into the body of the document. The same XSL stylesheet is used to display them as used with the original documents giving them a familiar look.


Document for Intellectual Property - Microsoft Internet Explorer
File
Edit
View
Favorites
Local intranet

From http://ivor.ecs.soton.ac.uk/porg2/xml/CORREO_Office_revisions_(approved_version).xml

With the Pannier coming through trials more effectively, the decision was to build an `Argentinean version' to overcome cost, shipping and copyright issues.

From http://ivor.ecs.soton.ac.uk/porg2/xm[illegible]_Modernising_Correo.xml

Containerisation

Whilst reviewing the transport, processing and delivery operations of Correo, it was also logical to incorporate containers into the way of working: essential for automation (which was limited in Correo), but also for WLA in Deliveries. After a lot of contact with UK experts in containerisation: manuals and IE studies were sent, the Argentines could clearly see the benefits and the focus was put on obtaining the `right' equipment to do it. With the proviso that trays and cages would be locally sourced from a design developed specifically for the purpose. So a specification was developed from:

The UK containerisation manuals and contacts.
Previous containerisation benchmarking to Sweden Post and La Poste
South American plastics manufacturers' exhibitions
Advise from [illegible]

From http://ivor.ecs.soton.ac.uk/porg2/xm[illegible]_Transcript.xml

A. Yes. It was innately obvious that we needed some containerisation or containerisation would be a benefit so we contacted the containerisation teams - [illegible] etc. back in the UK, got the manual; out there got it translated into Spanish and we was therefore able to show them the pictures of the equipment that we use and how we used it and able to show them the industrial engineering studies which proved the savings between a manual operation and a mechanised operation and a container operation and we tried to get their IE's involved in and doing some study work themselves to identify

Figure C.15: The Generated Document for the Sub-Area 'Intellectual Property'.

Figure C.15 shows one of the newly created documents. In this case the document covers the area of 'Intellectual Property' and includes paragraphs from a selection of documents in the set. Each included section starts with an explanation of its source and a link to that paragraph in the original source document. This allows users to read the extracts in their original context and read further if they wish. In the figure the names of people have been blurred out for confidentiality reasons.

C.8.1 The XLink Based XML Fragment Resolver

For this project a Link Resolver was developed to fully support parsing and loading a linkbase conforming to the final version of the XLink specification and to perform XSL transforms as instructed by the linkbase.

The destinations of the links in use are XPointer/XSL statements that the Resolver applies to an XML document by utilising an XML processor such as the Apache XML parser Xerces (Apache Software Foundation, 2001b). The XMLFragmentResolver returns the results of the XSL transformation back to the parent system. This is a departure from the first Resolver application which added links into HTML pages. This Resolver is designed to work with XML instead but as the specification for Resolvers is so loose the system needed little alteration to cope with this new usage. The Resolver does not return any information for the header and footer requests. The parser in the CA-DLS was modified to understand XML parsing as opposed to HTML parsing and be able to switch between them.

Another departure from the original reason for the development of the CA-DLS was that this Link Resolver was used both with a standalone program and within the CA-DLS. The Link Resolver still worked with the CA-DLS though the process of interpreting the transforms and extracting the XML paragraphs slowed the system to a degree that it became unusable.

To complete the project the 'Technical Map' and accompanying linkbase was batch processed by the KMReader Link Resolver harness to generate the finished documents on the various technical subjects. These were considered the final deliverables as demonstrators of the power of the approach. As well as generating new XML files the HTML versions were also generated making them more accessible to users of older Web browsers.

Because of the volume of material involved and the slow XPath engine the processing would take a number of minutes to run. It is not currently feasible to use the system dynamically with any useful speed. However, because the code is written into a Link Resolver it worked within the CA-DLS operating as a Web proxy. A user could click on a link in the 'Technical Knowledge Map' and the link service would generate a new document by applying all of the XPath transforms to all of the documents extracting the paragraphs. In a few minutes a new document that represented all the best knowledge on a certain topic would appear.

C.8.2 A Scenario for an Authoring System

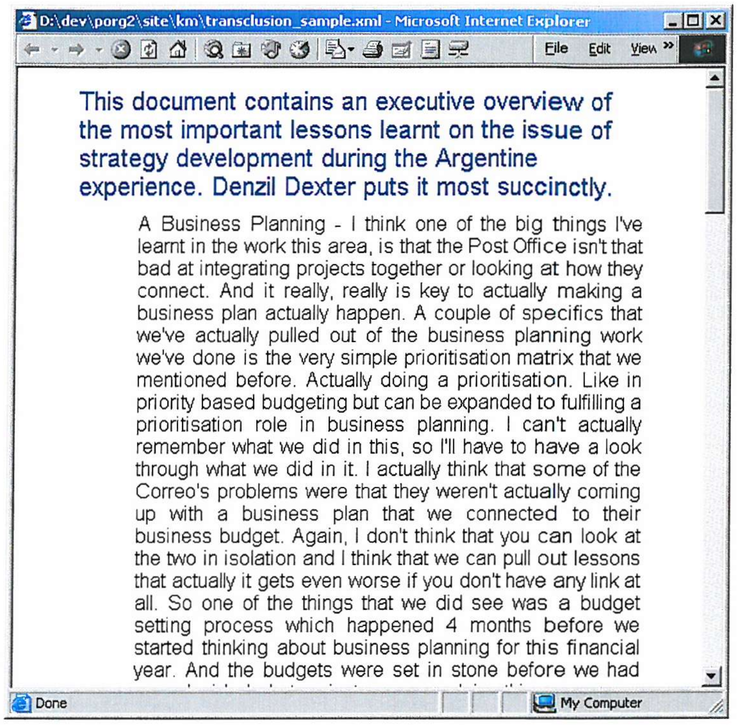
The technology described here could provide the foundation for a new document authoring tool. In the scenario an author writes an analysis report similar the one used as the source for this project. They use links to reference paragraphs from the source documents. A modified version of the KMReader could replace those links with the actual paragraphs from the original documents.

A simple XML document has been authored following an analysis of the document set. In it paragraphs from source documents are referred to using the id of a link in a linkbase. Users place special tags in the document to indicating which paragraph to use by using the link id. When this document is passed through the CA-DLS the service will call the Resolver with that link id and the Resolver will replace the reference with the paragraphs found at the destination or destinations of that link.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="../document.xsl" type="text/xsl"?>
<wp:Document
  xmlns:wp='http://webworks.com/publisher/xml/schema'
  title="Executive Issues concerned with Strategy Development">
<wp:Content>
<Heading2 wp:class="Heading2" wp:id="999263" wp:style="display: block;"
wp:type="para">
This document contains an executive overview of the most important lessons learnt
on the issue of strategy development during the Argentine experience.
Denzil Dexter puts it most succinctly.
</Heading2>
<paragraph id="4" />
</wp:Content>
</wp:Document>
```

Figure C.16: An Example of a New XML Document that an Author Could Create.

When this document is opened through the Muffin proxy using the XMLFragmentResolver <paragraph id=4/> will be replaced with the relevant paragraph pertaining to this subject as specified in the linkbase. A new document is easily created using simple references to the original document.


D:\dev\porg2\site\km\transclusion_sample.xml - Microsoft Internet Explorer
File
Edit
View
Done
My Computer

This document contains an executive overview of the most important lessons learnt on the issue of strategy development during the Argentine experience. Denzil Dexter puts it most succinctly.

A Business Planning - I think one of the big things I've learnt in the work this area, is that the Post Office isn't that bad at integrating projects together or looking at how they connect. And it really, really is key to actually making a business plan actually happen. A couple of specifics that we've actually pulled out of the business planning work we've done is the very simple prioritisation matrix that we mentioned before. Actually doing a prioritisation. Like in priority based budgeting but can be expanded to fulfilling a prioritisation role in business planning. I can't actually remember what we did in this, so I'll have to have a look through what we did in it. I actually think that some of the Correo's problems were that they weren't actually coming up with a business plan that we connected to their business budget. Again, I don't think that you can look at the two in isolation and I think that we can pull out lessons that actually it gets even worse if you don't have any link at all. So one of the things that we did see was a budget setting process which happened 4 months before we started thinking about business planning for this financial year. And the budgets were set in stone before we had

Figure C.17: The Document in Figure C.16 Rendered.

This is a simple example of the types of publishing scenario envisaged by a number of groups. For example the Apache Software Foundation has produced Cocoon (Apache Software Foundation, 2001a). A powerful way to organise the publishing of material for the Web. It uses the ability to produce a pipeline of XSL transformations to manage the publishing of XML data. (Wilde & Lowe, 2002, Wilde & Lowe, 2000) use these technologies to argue for a move towards a Web publishing model that is less concentrated on content publishing but involves a better balance application architecture using XLink as a key mechanism for a link based publishing platform.

C.9 Project Conclusions

This project provided an external industrial partner with a new way of thinking about how to write documents and manage the information within them. The system show-cased W3C open technologies and open hypermedia solutions to an audience used to dealing only with standard office software and Web pages. It was demonstrated that XML is a key new technology allowing information reuse and interchange in ways that have not been widely used before.

The deliverable was a Web based version of a report previously written in Microsoft Word. The report had used tables and tree diagrams to catalogue and categorise an analysis of the document set described in Appendix B. The major findings were displayed in tree diagrams. The detailed references were tables whose columns held the data for where to find pages in the document set considered to be worth reading on a specific topic. This high quality analysis work was the result of human labour and not machine analysis. The problem was that it was difficult to follow the references and the structured knowledge found in the diagrams was locked within the images. This project reproduced the report online using open hypermedia and open technologies to make the data reusable and to link the analysis back to the source documents in a powerful way.

There is a contrast with the previous project in that the former produced a working link server which would dynamically add links to documents. It was a stable product that could be distributed. This project produced a set of components and ideas to illustrate technologies and concepts. The end product was a set of static XML and HTML documents as well the data files such as the linkbase. The software was never brought to such a polished state. When the starting points for the two projects are considered the reverse is true. The first project was given fragments of data and a document set. The quality of the data was not as high as that in the second project and the resulting links between documents were subsequently less useful than they could have been. In the second project the starting point was a high quality analysis of the document set. The problem was that it was locked in a Microsoft Word document and diagrams. A dead end for the data.

An important issue is that a lot of information was lost in the capture process for two reasons. One was that the wrong questions had been asked and that the interviews need to be more structured to acquire the hard business facts that were desired of the project. The documents contained too much 'noise'. The second was that the transcripts and other documents used had no markup at all, there was a basic lack of competence in using styles in Microsoft Word. This restricts the options available to application writers hoping to reuse the data and is ultimately a considerable cost to the business. The contrast with the trend towards structured knowledge such as XML and RDF is quite acute. The problem lies in it being more expensive and difficult for busy people in the real world to produce documents with structured markup as opposed to just doing the utter minimum to get the report finished.

References

Akman, Varol, & Surav, Mehmet. 1996. Steps Toward Formalizing Context. AI Magazine, 17(3), 55–72.

Anderson, Kenneth M. 2000. Issues of data scalability in open hypermedia systems. New Review of Hypermedia, 151–178.

Andrews, Keith, Kappe, Frank, & Maurer, Hermann A. 1995. The Hyper-G Network Information System. The Journal of Universal Computer Science, 1(4), 206–220.

Apache Software Foundation. 2001a. Cocoon Project. http://xml.apache.org/cocoon/.

Apache Software Foundation. 2001b. Xerces Project. http://xml.apache.org/.

Barrett, R., & Maglio, P. P. 1999. Intermediaries: An approach to manipulating information streams. IBM Systems Journal, 38(4), 629–641.

Barrett, Rob, & Maglio, Paul P. 1998. Intermediaries: new places for producing and manipulating Web content. Computer Networks and ISDN Systems, 30(1–7), 509–518.

Bauer, Travis, & Leake, David B. 2001. WordSieve: A Method for Real-Time Context Extraction. Lecture Notes in Computer Science, 2116, 30–43.

Belkin, N. J., & Croft, W. B. 1992. Information Filtering and Information Retrieval: Two Sides of the Same Coin? Communications of the ACM, 35(12), 29–38.

Bernard, R., Crowder, R., Heath, I., & Hall, W. 1994 (Sept.). A Large-Scale Industrial Application of an Open Hypermedia System. In: Proceedings of the ACM Hypertext '94 Conference, Edinburgh, Scotland.

Berners-Lee, T., Hendler, J., & Lassila, O. 2001. The Semantic Web. Scientific American, May.

Bernstein, M. 1999. Where Are The Hypertexts? Hypertextual Bookmaking: Betting on the Future of Literature. Pages 1–1 of: Tochtermann, Klaus, Westbomke, Jörg, Wiil, Uffe K., & Leggett, John J. (eds), Proceedings of the ACM Hypertext '99 Conference, Darmstadt, Germany. New York, N.Y.: ACM Press.

Bollacker, Kurt D., Lawrence, Steve, & Giles, C. Lee. 1998. CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications. Pages 116–123 of: Sycara, Katia P., & Wooldridge, Michael (eds), Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98). New York: ACM Press.

Boyns, Mark. 2000. Muffin. A World Wide Web Filtering System. http://muffin.doit.org/.

Bra, P. De, Houben, G-J., & Wu, H. 1999. AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. Pages 147–156 of: Proceedings of the ACM Hypertext '99 Conference, Darmstadt, Germany.

Bra, Paul De, Aerts, Ad, Smith, David, & Stash, Natalia. 2002 (June). AHA! The Next Generation. In: Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia, University of Maryland, College Park, Maryland, USA.

Brin, Sergey, & Page, Lawrence. 1998. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 30(1–7), 107–117.

Brusilovsky, P., & Schwarz, E. 1997. User as Student: Towards an Adaptive Interface for Advanced Web-Based Applications. Pages 177–188 of: Jameson, Anthony, Paris, Cécile, & Tasso, Carlo (eds), Proceedings of the 6th International Conference on User Modeling (UM-97). CISM, vol. 383. Wien: Springer.

Brusilovsky, P., Eklund, J., & Schwarz, E. 1998 (Apr.). Web-based education for all: A tool for developing adaptive courseware. Pages 291–300 of: Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia.

Brusilovsky, Peter. 1996. Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 6(2-3), 87–129.

Brusilovsky, Peter, & Pesin, Leonid. 1995. Visual Annotation of Links in Adaptive Hypermedia. Pages 222–223 of: Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. Short Papers: Agents and Anthropomorphism, vol. 2.

Bumbulis, P. J., Cowan, D. D., Durance, C. M., & Stepien, T. M. 1993. An introduction to the OSI Directory Services. Computer Networks and ISDN Systems, 26(2), 239–249.

Bush, V. 1945. As We May Think. Atlantic Monthly, July, 101–108.

Carr, L. A., Hall, W., Davis, H.C, DeRoure, D., & R., Hollom. 1994 (May). The Microcosm Link Service and its Application to the World Wide Web. In: Proceedings of the First International World Wide Web Conference, Geneva, Switzerland.

Carr, Les A., DeRoure, David C., Davis, Hugh C., & Hall, Wendy. 1998. Implementing an Open Link Service for the World Wide Web. World Wide Web Journal, 1(2).

Chen, Chaomei. 1999. Information Visualization and Virtual Environments. Springer.

Cheverst, Keith, Davies, Nigel, Mitchell, Keith, & Friday, Adrian. 2000. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. Pages 20–31 of: Mobile Computing and Networking.

Clark, J. 1999. XSL Transformations (XSLT), Version 1.0. W3C Recommendation http://www.w3.org/TR/xslt.

Clark, J., & Deach, S. Aug 18, 1998. Extensible Stylesheet Language (XSL), Version 1.0. W3C Recommendation http://www.w3.org/TR/WD-xsl.

Clark, J., & DeRose (Eds), S. J. 1999 (Apr.). "XML Path Language (XPath) Version 1.0". W3C Recommendation http://www.w3.org/TR/xpath.

Conklin, J. 1987. Hypertext: An Introduction and Survey. IEEE Computer, 20(9), 17–40.

Davies, Nigel, Mitchell, Keith, Cheverst, Keith, & Blair, Gordon. 1998. Developing a Context Sensitive Tourist Guide. Page 10 of: Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices.

Davis, Fred D. 1993. User Acceptance of Information Technology: System Characteristics, User Perceptions and Behavioral Impacts. International Journal of Man-Machine Studies, 38(3), 475–487.

Davis, Fred D., & Venkatesh, Viswanath. 1996. A Critical Assessment of Potential Measurement Biases in the Technology Acceptance Model: Three Experiments. International Journal of Human-Computer Studies, 45(1), 19–45.

Davis, H. C. 1995 (Nov.). Data Integrity Problems in an Open Hypermedia Link Service. Ph.D. thesis, Department of Electronics and Computer Science, University of Southampton.

Davis, H. C., Knight, S. J., & Hall, W. 1994 (Sept.). Light Hypermedia Services: A Study of Third Party Application Integration. Pages 41–50 of: Proceedings of the ACM Hypertext '94 Conference, Edinburgh, Scotland.

Davis, H. C., Lewis, A. J., & Rizk, A. 1996. OHP: A Draft Proposal for a Standard Open Hypermedia Protocol. Pages 27–53 of: Wiil, Uffe Kock, & Demeyer, Serge (eds), Proceedings of the 2nd Workshop on Open Hypermedia Systems, ACM Hypertext '96. Washington, D.C.: Available as Report No. ICS-TR-96-10 from the Dept. of Information and Computer Science, University of California, Irvine.

de Bra, Paul, & Calvi, Licia. 1998. AHA! An open adaptive hypermedia architecture. New Review of Hypermedia and Multimedia, 4, 115–139.

Delisle, Norman M., & Schwartz, Mayer D. 1987. Contexts — A Partitioning Concept for Hypertext. ACM Transactions on Office Information Systems, 5(2), 168–186. Special Issue on Computer-Supported Cooperative Work.

Dennett, Daniel C. 1987. Cognitive Wheels: The Frame Problem of AI. Pages 41–64 of: Pylyshyn, Zenon (ed), The Robot's Dilemma: The Frame Problem in Artificial Intelligence. Norwood, New Jersey: Ablex Publishing Co.

DeRose, S. J., Maler, E., & Orchard (Eds), D. 2000 (Dec.). XML Linking Language (XLink) Version 1.0. W3C Proposed Recommendation http://www.w3.org/TR/xlink/.

DeRose, S. J., Maler, E., & Jr. (Eds), R. Daniel. 2001 (Sept.). XML Pointer Language (XPointer) Version 1.0. W3C Candidate Recommendation http://www.w3.org/TR/xptr/.

Dervin, B. 1997. Given a context by any other name: Methodological tools for taming the unruly beast. Pages 13–38 of: Vakkari, P., Savolainen, R., & Dervin, B. (eds), Information seeking in context. London, UK: Taylor Graham.

Dey, Anind K. 2000 (Nov.). Providing Architectural Support for Building Context-Aware Applications. Ph.D. thesis, Georgia Institute of Technology.

Dey, Anind K., Salber, Daniel, & Abowd, Gregory D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human Computer Interaction (HCI) Journal, 16(2-4), 97–166.

Edmonds, B. 1999. The Pragmatic Roots of Context. Pages 119–132 of: Bouquet, Paolo, Serafini, Luciano, Brézillon, Patrick, Benerecetti, Massimo, & Castellani, Francesca (eds), Proceedings of the 2nd International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-99). LNAI, vol. 1688. Berlin: Springer.

El-Beltagy, Samhaa, Hall, Wendy, Roure, David De, & Carr, Leslie. 2001 (Aug.). Linking in Context. Pages 151–160 of: Proceedings of the ACM Hypertext 2001 Conference, Aarhus, Denmark.

Englebart, D. C. 1963. A Conceptual Framework for the Augmentation of Man's Intellect. Vistas of Information Handling, 1.

Foltz, Peter W., & Dumais, Susan T. 1992. Personalized Information Delivery: An Analysis of Information-Filtering Methods. Communications of the ACM, 35(12), 51–60.

Fountain, A., Hall, W., Heath, I., & Davis, H. C. 1990 (Nov.). Microcosm: An Open Model for Hypermedia With Dynamic Linking. Pages 298–311 of: Rizk, A., Streitz, N., & André, J. (eds), Hypertext: Concepts, Systems and Applications, Proceedings of the Hypertext '90 Conference, INRIA, France.

Fox, A., & Brewer, E. A. 1996. Reducing WWW latency and bandwidth requirements by real-time distillation. Computer Networks and ISDN Systems, 28(7-11), 1445–56.

Garfield, Eugene. 1979. Citation Indexing: Its Theory and Application in Science, Technology, and Humanities. New York: Wiley.

Grønbæk, K., Hem, J. A., Madsen, O. L., & Sloth, L. 1994. Cooperative Hypermedia Systems: A Dexter-Based Architecture. Communications of the ACM, 37(2), 64–75.

Grønbæk, K., Bouvin, N. O., & Sloth, L. 1997 (Apr.). Designing Dexter-based Hypermedia Systems for the World Wide Web. Pages 146–156 of: Proceedings of the ACM Hypertext '97 Conference, Southampton, UK.

Gruber, Thomas R. 1993 (Aug.). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Tech. rept. KSL-93-04. Knowledge Systems Laboratory, Stanford University.

Haan, B. J., Kahn, P., Riley, V. A., Coombs, J. H., & Meyrowitz, N. 1992. IRIS Hypermedia Services. Communications of the ACM, 35(1), 36–51.

Halasz, F. G., & Schwartz, M. 1994. The Dexter Hypertext Reference Model. Communications of the ACM, 31(2), 30–39.

Hall, W., Davis, H. C., & Hutchings, G. A. 1996. Rethinking Hypermedia: The Microcosm Approach. Kluwer Academic Press.

Hardman, L., Bulterman, D. C. A., & Van Rossum, G. 1994. Adding Time and Content to the Dexter Model. Communications of the ACM, 37(2), 50–63.

Hardman, Lynda, Bulterman, Dick C. A., & Rossum, Guido Van. 1993. Links in Hypermedia: the Requirement for Context. Pages 183–191 of: Proceedings of the ACM Hypertext '93 Conference, Seattle, Washington, USA. New York, NY, USA: ACM Press.

Hardman, Lynda, van Ossenbruggen, Jacco, Mullender, K. Sjoerd, Rutledge, Lloyd, & Bulterman, Dick C. A. 1999. Do You Have the Time? Composition and Linking in Time-Based Hypermedia. Pages 189–196 of: Proceedings of the Tenth ACM Conference on Hypertext.

Harter, Andy, Hopper, Andy, Steggles, Pete, Ward, Andy, & Webster, Paul. 2002. The Anatomy of a Context-Aware Application. Wireless Networks, 9(Mar.), 187–197.

Hawgood, David. 1998. GEDCOM Data Transfer. Federation of Family History Societies (Publications). ISBN 094815117X.

Hill, G. J., Wilkins, R. J., & Hall, W. 1993. Open and Reconfigurable Hypermedia Systems: A Filter Based Model. Hypermedia, 5(2), 103–118.

Hitchcock, S., Carr, L., Harris, S., Hey, J. M. N., & Hall, W. 1997. Citation linking: Improving access to online journals. Pages 115–122 of: Allen, Robert B., & Rasmussen, Edie (eds), Proceedings of the 2nd ACM International Conference on Digital Libraries. New York: ACM Press.

Hitchcock, Steve, & Hall, Wendy. 2001. How Dynamic E-journals can Interconnect Open Access Archives. Pages 183–193 of: Hubler, Arved, Linde, Peter, & Smith, John W. T. (eds), Electronic Publishing '01: 2001 in the Digital Publishing Odyssey. Amsterdam: IOS Press, for ICCC/IFIP.

Hitchcock, Steve, Carr, Les, Hall, Wendy, Harris, Steve, Probets, Steve, Evans, David, & Brailsford, David. 1998 (Dec. 15,). Linking electronic journals: Lessons from the Open Journal project. Technical Report. D-Lib Magazine.

Hohl, Hubertus, Boecker, Heinz-Dieter, & Gunzenhaeuser, Rul. 1996. Hypadapter: An adaptive hypertext system for exploratory learning and programming. User Modelling and User-Adapted Interaction, 6(2-3), 131–156.

Hothi, J, & Hall, W. 1998. An Evaluation of Adapted Hypermedia Techniques Using Static User Modelling. Pages 45–50 of: Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, USA.

Hothi, Jatinder, Hall, Wendy, & Sly, Tim. 2000. A Study Comparing the Use of Shaded Text and Adaptive Navigational Support in Adaptive Hypermedia. Lecture Notes in Computer Science, 1892, 335–342.

Hughes, Gareth, & Carr, Leslie. 2002 (June). Microsoft Smart Tags: Support, Ignore or Condemn Them? In: Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia, University of Maryland, College Park, Maryland, USA.

Kacmar, C. J., & Leggett, J. J. 1991. A Process-Oriented Extensible Hypertext Architecture. ACM Transaction on Information Systems, 9(4), 399–419.

Kindberg, Tim, Barton, John, Morgan, Jeff, Becker, Gene, Caswell, Debbie, Debaty, Philippe, Gopal, Gita, Frid, Marcos, Krishnan, Venky, Morris, Howard, Schettino, John, Serra, Bill, & Spasojevic, Mirjana. 2000. People, Places, Things: Web Presence for the Real World. Tech. rept. Hewlett Packard Labs.

Knight, S. J. 1996 (May). Abstracting Anchors from Documents. Ph.D. thesis, Department of Electronics and Computer Science, University of Southampton.

Lawrence, Steve, Giles, C. Lee, & Bollacker, Kurt. 1999. Digital Libraries and Autonomous Citation Indexing. IEEE Computer, 32(6), 67–71.

Leggett, J. J., & Schnase, J. L. 1994. Dexter With Open Eyes. Communications of the ACM, 37(2), 77–86.

Lieberman, H., & Selker, T. 2000. Out of context: Computer systems that adapt to, and learn from, context. IBM Systems Journal, 39(3/4), 617–632.

Loeb, Shoshana, & Terry, Douglas. 1992. Information filtering. Communications of the ACM, 35(12), 26–28.

Lowe, D., Hall, W., & Ginige, A. 1997. Principles of Hypermedia Engineering: Beyond the Web. Wiley and Sons, Inc.

Lueg, Christopher. 2002. On the Gap Between Vision and Feasibility. In: International Conference on Pervasive Computing. Springer.

Malcolm, K. C., E., Poltrock S., & D., Schuler. 1991 (Dec.). Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise. Pages 13–25 of: Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA.

Mankoff, Jennifer, & Schilit, Bill N. 1997. Supporting Knowledge Workers Beyond the Desktop with Palplates. Pages 550–551 of: Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems. TECHNICAL NOTES: Beyond the Desktop, vol. 1.

Maurer, H. 1995. Hyper-G now Hyperwave: the next generation Web solution. Addison Wesley.

McCarthy, John. 1987. Generality in artificial intelligence. Communications of the ACM, 30(12), 1030–1035.

Meyrowitz, N. 1989. The Missing Link: Why We're All Doing Hypertext Wrong. The Society of Text, MIT Press, Cambridge, Massachusetts, 107–114.

Michaelides, Danius T., Millard, David E., Weal, Mark J., & Roure, David C. De. 2001. Auld Leaky: A Contextual Open Hypermedia Link Server. In: Proceedings of the 7th Workshop on Open Hypermedia Systems, ACM Hypertext 2001 Conference.

Miles-Board, Timothy, & Carr, Leslie. 2002 (June). Looking for Linking: Associative Links on the Web. In: Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia, University of Maryland, College Park, Maryland, USA.

Millard, Dave, Moreau, Luc, Davis, Hugh, & Reich, Sigi. 2000. FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains. In: Proceedings of the Hypertext Conference HT'00.

Moreau, Luc, Gibbins, Nick, DeRoure, David, El-Beltagy, Samhaa, Hall, Wendy, Hughes, Gareth, Joyce, Dan, Kim, Sanghee, Michaelides, Danius, Millard, Dave, Reich, Sigi, Tansley, Robert, & Weal, Mark. 2000. SoFAR with DIM Agents: An Agent Framework for Distributed Information Management. In: Proceedings of the Fifth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems.

Mukherjea, Sougata, & Hara, Yoshinori. 1997. Focus+Context Views of World-Wide Web Nodes. Pages 187–196 of: Proceedings of the ACM Hypertext '97 Conference, Southampton, UK. Visualization.

Mynatt, Elizabeth D., Back, Maribeth, Want, Roy, & Frederick, Ron. 1997. Audio Aura: Light-Weight Audio Augmented Reality. Pages 211–212 of: Proceedings of the ACM Symposium on User Interface Software and Technology. Blurring Physical and Virtual.

Nelson, T. H. 1987. Literary Machines. Mindful Press.

Neumüller, Moritz. 2000 (Apr.). A Semiotic Analysis of iMarketing Tools. Pages 238–239 of: Proceedings of the ACM Hypertext 2000 Conference, San Antonio, Texas, USA.

Nielsen, J. 1989. Usability Engineering at a Discount. Vol. Designing and using Human-Computer Interfaces and Knowledge Based Systems. Editors, Salvengy G & Smith MJ. Elsevier. Pages 394–401.

Nielsen, J. 1994. Enhancing the Explanatory Power of Usability Heuristics. Page 210 of: Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems. PAPER ABSTRACTS: Tools for Design, vol. 2.

Østerbye, K., & Wiil, U. K. 1996 (Mar.). The Flag Taxonomy of Open Hypermedia Systems. Pages 129–139 of: Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA.

Pascoe, Jason, Ryan, Nick, & Morse, David. 1998. Human-Computer-Giraffe Interaction: HCI in the Field. Page 8 of: Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices.

Pearl, Amy. 1991 (Nov.). Sun's Link Service: A Protocol for Open Linking. Pages 137–146 of: Proceedings of the Hypertext '89 Conference on Hypertext, Pittsburgh, Pennsylvania, USA.

Polson, Martha C., & Richardson, J. Jeffrey (eds). 1988. Foundations of Intelligent Tutoring Systems. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Preece, Jenny, Rogers, Yvonne, Sharp, Helen, Benyon, David, Holland, Simon, & Carey, Tom. 1994. Human-Computer Interaction. Reading, Mass.: Addison-Wesley Publishing.

Reich, Siegfried, Griffiths, Jon, Millard, David E., & Davis, Hugh C. 1999. Solent - A Platform for Distributed Open Hypermedia Applications. Pages 802–811 of: Database and Expert Systems Applications.

Reich, Siegfried, Wiil, Uffe K., Nürnberg, Peter J., Davis, Hugh C., Grønbæk, Kaj, Anderson, Kenneth M., Millard, David E., & Haake, Jörg M. 2000. Addressing Interoperability in Open Hypermedia: The Design of the Open Hypermedia Protocol. New Review of Hypermedia, 207–243.

Rizk, A., & Sauter, I. 1992 (Nov.). Multicard: An Open Hypermedia System. Pages 4–10 of: Lucarella, D., Nanard, M., J., Nanard, & Paolini, P. (eds), Proceedings of the ACM Hypertext '92 Conference, Milano, Italy.

S. J. DeRose (Editor). 1999. XML XLink Requirements Version 1.0. W3C Note http://www.w3.org/TR/1999/NOTE-xlink-req-19990224/.

Salber, Daniel, Dey, Anind K., & Abowd, Gregory D. 1999. The Context Toolkit: Aiding the Development of Context-Enabled Applications. Pages 434–441 of: Williams, Marian G., Altom, Mark W., Ehrlich, Kate, & Newman, William (eds), Proceedings of the Conference on Human Factors in Computing Systems (CHI-99). New York: ACM Press.

Salton, Gerard, & Buckley, Christopher. 1988. Term-weighting approaches in automatic text retrieval. Information Processing and Management, 24(5), 513–523.

Schilit, Bill, Adams, Norman, & Want, Roy. 1994 (Dec.). Context-Aware Computing Applications. In: IEEE Workshop on Mobile Computing Systems and Applications. Palo Alto Research Center, Santa Cruz, CA.

Schilit, Bill N. 1995 (May). A System Architecture for Context-Aware Mobile Computing. Ph.D. thesis, Columbia University. http://sandbox.parc.xerox.com/parctab/.

Schnase, John L., Leggett, John J., Hicks, David L., Nuernberg, Peter J., & Sánchez, J. Alfredo. 1993. Design and Implementation of the HB1 hyperbase management system. Electronic Publishing Origination, Dissemination and Design, 6(1), 35–63.

Selker, T., & Burleson, W. 2000. Context-aware design and interaction in computer systems. IBM Systems Journal, 39(3/4), 880–891.

Turner, Roy M. 1997. Context-Mediated Behavior for Intelligent Agents. International Journal of Human-Computer Studies.

Turner, Roy M. 1998. Context-Mediated Behavior for AI Applications. Pages 538–545 of: IEA/AIE (Vol. 1).

van Ossenbruggen, Jacco, Geurts, Joost, Cornelissen, Frank, Rutledge, Lloyd, & Hardman, Lynda. 2001. Towards Second and Third Generation Web-Based Multimedia. Pages 479–488 of: WWW2001.

van Ossenbruggen, Jacco, Hardman, Lynda, & Rutledge, Lloyd. 2002. Hypermedia and the Semantic Web: A Research Agenda. Journal of Digital Information, 3(1).

W3C Consortium. 1998a. Extensible Markup Language (XML) 1.0 Specification. http://www.w3.org/TR/REC-xml.

W3C Consortium. 1998b. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. http://www.w3.org/TR/REC-smil.

W3C Consortium. 2001. Synchronized Multimedia Integration Language (SMIL) 2.0. http://www.w3.org/TR/REC-smil20.

Weal, Mark J., Hughes, Gareth V., Millard, David E., & Moreau, Luc. 2001 (Aug.). Open Hypermedia as a Navigational Interface to Ontological Information Spaces. Pages 227–236 of: Proceedings of the Twelveth ACM Conference on Hypertext and Hypermedia HT'01.

Weiser, M. 1991. The Computer for the Twenty-First Century. Scientific American, 265(3), 94–104.

Weiser, M. 1993. Some computer science issues in ubiquitous computing. Communications of the ACM, 36(7), 74–84.

Wiil, Uffe Kock, & Nurnberg, Peter J. 1999. Evolving Hypermedia Middleware Services: Lessons and Observations. Pages 427–436 of: ACM Symposium on Applied Computing (SAC '99).

Wilde, Erik, & Lowe, David. 2000. From Content-Centered Publishing to a Link-based View of Information Resources. In: Proceedings of the 33rd Hawaii International Conference on System Sciences. IEEE Computer Society Press.

Wilde, Erik, & Lowe, David. 2002. XPath, XLink, XPointer, and XML: A Practical Guide to Web Hyperlinking and Transclusion. Reading, Massachusetts: Addison Wesley.

Wills, G.B, Heath, I, Crowder, R.M, & Hall, W. 1997. Evaluation of a User Interface Developed for Industrial Applications. Tech. rept. M97-4, ISBN-0854326499. University of Southampton.

Wills, G.B, Hughes, G.V., & Hall, W. 1999 (November). Evaluation of AIMS-Academic Information Management System. Tech. rept. M99-5, ISBN-085432700-2. University of Southampton.

Wolber, D., Kepe, M., & Ranitovic, I. 2002. Exposing Document Context in the Personal Web. Pages 151–158 of: Proceedings of the 7th ACM International Conference on Intelligent User Interfaces.

Wu, H., & Bra, P. De. 2002 (May). Link-Independent Navigation Support in Web-Based Adaptive Hypermedia. In: Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii, USA.

Yan, H., & Selker, T. 2000 (Jan.). Context Aware Office Assistant. Pages 276–279 of: Proceedings of the 2000 International Conference on Intelligent User Interfaces, New Orleans, LA USA.

Yankelovich, N., Haan, B. J., Meyrowitz, N., & Drucker, S. M. 1988. Intermedia: The Concept and the Construction of a Seamless Information Environment. IEEE Computer, 21(1), 81–96.

Yeong, Wengyik, Howes, Tim, & Kille, Steven. 1993 (July). Lightweight Directory Access Protocol. Internet proposed standard RFC 1487.