University of Southampton

A Framework for Distributed Open Hypermedia

by

Stuart Goose

A thesis submitted for the degree of

Doctor of Philosophy

in the

Faculty of Engineering and Applied Science

Electronics and Computer Science

July 22, 1997

University of Southampton

ABSTRACT

Faculty of Engineering and Applied Science

Electronics and Computer Science

Doctor of Philosophy

A Framework for Distributed Open Hypermedia

by Stuart Goose

Early hypermedia systems are now referred to as *closed* systems due to their reliance upon proprietary document formats, embedded links, monolithic designs and their inability to integrate with the desktop environment. Advances from the research community gave rise to a new generation of *open* hypermedia systems. Open hypermedia systems tend to have modular architectures and can offer an extensible hypermedia link service integrated with the desktop environment.

One explanation for the unmitigated success of the World Wide Web (WWW) is the way in which common Internet services have been integrated. Although the WWW is open in terms of the platforms and protocols supported, it has several limitations in common with closed hypermedia systems. Described in this thesis is Microcosm TNG, a framework for distributed open hypermedia, able to integrate with both the desktop and the Internet.

The abstraction of a *hypermedia application* is introduced for improving the organisation and the encapsulation of related documents and link services. Such a resource can then be made available to remote users via the Microcosm TNG framework. Architectural extensions to the system are described which enable the hierarchical composition of hypermedia applications to construct rich distributed information spaces.

Contents

1 Introduction **13**

1.1 The Origins of Hypertext 13

1.2 Characterising Hypertext and Hypermedia 13

1.3 Defining Hypermedia 14

1.4 Distributed Hypermedia 15

1.5 Thesis Structure 15

1.6 Declaration 16

2 The Importance of Being "Open" **17**

2.1 The Open Systems Movement 17

2.2 The Tide of Change 17

2.3 The Dexter Reference Model 21

2.4 Impetus from Industry 24

2.5 Various Approaches to Delivering Open Hypermedia 28

2.5.1 Scripting Languages 28

2.5.2 The Toolkit Approach 28

2.5.3 Link Services 29

2.5.4 Hyperbases 30

2.6 Hypermedia Standards 31

2.6.1 HyTime 31

2.6.2 MHEG 32

2.6.3 HTML 33

2.6.4 HTTP 33

2.7 Hypermedia Interchange 33

2.8 Distilling the Requirements for "Open" Hypermedia 34

2.8.1 A Definition 35

2.8.2 A Conceptual Model 38

2.8.3 A Protocol . . . 40

2.9 Are There Disadvantages to Openness ? . . . 40

2.9.1 The Editing Problem . . . 40

2.9.2 The Versioning Problem . . . 41

2.10 Summary . . . 41

3 Taxonomic Review of Open Hypermedia Systems **43**

3.1 Introduction . . . 43

3.2 World Wide Web (WWW) . . . 43

3.3 Intermedia . . . 46

3.4 Sun Link Service . . . 48

3.5 Multicard . . . 50

3.6 PROXHY . . . 51

3.7 Chimera . . . 53

3.8 Hyper-G . . . 55

3.9 DeVise Hypermedia (DHM) . . . 56

3.10 System Prototype 3 (SP3) . . . 58

3.11 HyperDisco . . . 60

3.12 Summary . . . 61

4 Microcosm: A Model for Open Hypermedia **64**

4.1 Project Aims . . . 64

4.2 The Conceptual Model of Microcosm . . . 66

4.3 The Microcosm Architecture . . . 66

4.3.1 Loosely Coupled Independent Processes . . . 67

4.3.2 Inter-Process Communication . . . 68

4.3.3 Extensible Message Format . . . 69

4.4 The Document Control System (DCS) . . . 70

4.5 The Filter Management System (FMS) . . . 70

4.6 Document Viewers . . . 71

4.6.1 Fully Aware Viewers . . . 71

4.6.2 Semi-Aware Viewers . . . 72

4.6.3 Unaware Viewers . . . 73

4.7 Service Providing Filters . . . 74

4.7.1 Linker Filter . . . 74

4.7.2 Linkbase Filter . . . 75

4.7.3 ODBC SQL Filter . . . 77

4.7.4 Computed Linker Filter . . . 78

4.7.5 Navigation Filters . . . 79

4.7.6 Available Links Filter . . . 79

4.8 Support for Multimedia . . . 80

4.9 The Document Management System . . . 81

4.10 System Configuration . . . 82

4.11 Application Areas of Microcosm . . . 82

4.11.1 Applied to Education . . . 82

4.11.2 Applied to Engineering . . . 83

4.12 Summary . . . 84

5 Distributed Information Systems 86

5.1 Introduction . . . 86

5.2 Distributed Information . . . 86

5.3 Distributed Resource Discovery Systems . . . 87

5.3.1 Archie . . . 89

5.3.2 Wide Area Information Server (WAIS) . . . 89

5.3.3 Gopher . . . 90

5.3.4 Harvest . . . 90

5.4 The Vision of a Global Information System . . . 91

5.5 Hypermedia Systems with Local Distribution . . . 92

5.5.1 Neptune/HAM . . . 92

5.5.2 Knowledge Management System (KMS) . . . 93

5.5.3 Intermedia . . . 93

5.5.4 Sun Link Service . . . 93

5.6 Hypermedia Systems with Global Distribution . . . 94

5.6.1 World Wide Web (WWW) . . . 95

5.6.2 Hyper-G . . . 96

5.6.3 Virtual Notebook System (VNS) 98
5.6.4 Distributed Hypertext (DHT) 99
5.6.5 Multicard 99
5.6.6 Microcosm 100
5.6.7 HyperDisco 102
5.7 Distributed Link Services Above The WWW 103
5.7.1 Distributed Link Service 103
5.7.2 Aquarelle 103
5.7.3 DeVise Hypermedia (DHM) 104
5.7.4 Chimera 104
5.8 Summary 105

6 The Design of an Open and Customisable Distributed Communication Layer 108
6.1 Introduction 108
6.2 Issues in Distribution 108
6.2.1 The Advantages of Distributed Systems 109
6.2.2 The Disadvantages of Distributed Systems 109
6.2.3 Design Issues 109
6.2.4 The Importance of Communication 110
6.3 Analysis of the Models of Communication for Microcosm 110
6.3.1 Segregated Communication Model (SCM) 110
6.3.2 The Active FMS 114
6.3.3 The Direct Communication Model (DCM) 115
6.3.4 Summary 118
6.4 Additional Communication Issues 119
6.5 Review of Existing Distributed Technologies 119
6.6 The Design of the Heterogeneous Communication Model (HCM) . . 121
6.6.1 Process Addressing Scheme 121
6.6.2 Customising the Addressing Scheme 122
6.6.3 Process Registration 123
6.6.4 Sending a Message 123

6.6.5 Location Transparency 124
6.7 Architecture Of The HCM 124
6.8 Extending The HCM to Support Inter-Session Communication . . . 126
6.8.1 Extending the Custom Addressing Template 127
6.8.2 Message Router Negotiation 128
6.8.3 Providing Scalable Process Registration 128
6.9 Alternative HCM Configurations 132
6.10 Status of the HCM 133
6.11 Summary 133

7 Microcosm TNG: A Model for Distributed Open Hypermedia 135

7.1 Introduction 135
7.2 Requirements for Distributed Open Hypermedia 135
7.3 Architectural Constraints Upon Microcosm 140
7.4 Redefining the Notion of a Hypermedia Application 142
7.5 The Conceptual Model of Microcosm TNG 143
7.6 The Architecture of Microcosm TNG 144
7.6.1 Modelling Microcosm Using Microcosm TNG 145
7.6.2 Process-driven Infrastructure 145
7.6.3 Managing Dynamic Distributed Processes 145
7.6.4 Microcosm TNG Custom Addressing Template 146
7.7 Distributed User Sessions 146
7.7.1 A Distributed Single User Session 147
7.7.2 Multi-user Distributed Sessions 147
7.7.3 Comparison with Extended Microcosm 151
7.8 Hypermedia Services 153
7.8.1 Linker 154
7.8.2 Linkbase 155
7.8.3 Available Links 156
7.9 Document Management Services 158
7.9.1 Is a Document Management System Necessary ? 158
7.9.2 Limitations of the Microcosm DMS 158

7.9.3 Distributed Document Management 160
7.10 Presentation Services 166
7.10.1 What the User has to Face 166
7.10.2 Viewer Integration 167
7.10.3 Heterogeneous User Interfaces 169
7.11 Summary 170

8 A Reflexive Model for the Construction of Hypermedia Applications 173
8.1 Introduction 173
8.2 The Resource Based Approach to Hypermedia 173
8.3 Structuring Hypermedia Applications 174
8.3.1 Links 175
8.3.2 Link Services 178
8.4 Should Dexter be Extended ? 178
8.5 The Reflexive Model 179
8.5.1 Information Reuse 179
8.5.2 Reflexive Hypermedia Applications 180
8.6 Architecture to Support Reflexive Hypermedia Applications 182
8.6.1 Hypermedia Applications as Process Groups 182
8.6.2 Reflector Process 183
8.7 Hierarchies of Hypermedia Applications 184
8.7.1 Achieving Asynchrony in the Browser 186
8.8 Application Areas 186
8.8.1 Pirelli/FIRM 187
8.8.2 SToMP 188
8.8.3 Discussion 188
8.9 Limitations of the Implementation 189
8.10 Summary 190

9 Future Work and Conclusions 192
9.1 Future Work 192
9.1.1 Improving the HCM 192
9.1.2 Enhancing Microcosm TNG 193

9.1.3 Emerging Standards 196
9.2 Conclusions 196
9.2.1 The Open Approach 196
9.2.2 Distributed Hypermedia 197
9.2.3 Distributed Open Hypermedia 197

Bibliography **200**

List of Figures

2.1	The layers of the Dexter Model.	22
2.2	The Flag Taxonomy.	39
2.3	The broad categories distinguishable using the Flag Taxonomy.	39
2.4	Summary of the directions of research into open hypermedia.	41
3.1	Intermedia architecture.	47
3.2	Sun Link Service architecture.	49
3.3	Multicard architecture.	50
3.4	PROXHY architecture.	52
3.5	Chimera architecture.	54
3.6	System Prototype 3 architecture.	59
3.7	HyperDisco architecture.	60
4.1	The conceptual model of Microcosm.	66
4.2	The Microcosm architecture.	67
4.3	Applying an action to a selection within a fully aware Microcosm viewer.	72
4.4	Making hypermedia services available from within an unaware application by adding functional icons to the title bar.	73
4.5	The internal representation of a (a) specific link (b) local link (c) generic link in Microcosm.	76
4.6	The Microcosm viewer for sound.	80
4.7	The three levels at which Microcosm can be configured.	82
5.1	The rapid growth in the number of Internet hosts. Source: Network Wizards (http://www.nw.com)	88
5.2	The rapid growth in the number of Internet domains. Source: Network Wizards (http://www.nw.com)	88

5.3 The Hyper-G architecture. 97
5.4 The Multicard architecture. 100
5.5 The range of Microcosm configurations supported by Hill. 101

6.1 The Segregated Communication Model. 111
6.2 The Active Filter Management System. 114
6.3 The Direct Communication Model. 116
6.4 The logical model of a single user session. 125
6.5 One possible physical realisation of a single user session. 125
6.6 The layered construction of the HCM. 127
6.7 The HCM providing a widely distributed multi-user environment. . 133

7.1 The relationship between users, sessions and hypermedia applications. 143
7.2 Combining the three principal facets of an Open Hypermedia System. 144
7.3 The logical model of a Microcosm TNG single user session. 147
7.4 One possible physical realisation of a Microcosm TNG single user session. 148
7.5 A screenshot of a utility for browsing and connecting to remote hypermedia applications. 149
7.6 Microcosm TNG users sharing hypermedia applications in a distributed environment. 150
7.7 Extending Microcosm to facilitate a distributed filter chain. 151
7.8 An alternative distributed architecture of Microcosm TNG. 152
7.9 The Available Links process supporting multiple asynchronous queries. 157
7.10 A viewer showing two selected applications in the menu. 166
7.11 The construction of a shortcut using the tngtool. 167
7.12 Creating a link under Microcosm TNG using Microsoft Word and Emacs. 168
7.13 The user interface to the Microcosm TNG distributed DMS browser. 169

8.1 Organising link storage along the dimensions of context and location. 177

8.2 A hierarchy of reflexive Microcosm TNG hypermedia applications. . . 181
8.3 The *Cats* hypermedia application is enriched by the *Lions* and *Tigers* applications. 182
8.4 The abstraction of a hypermedia application under Microcosm TNG. 184
8.5 The distributed DMS browser showing a unified view of the logical document structure of the *Cats* hypermedia application hierarchy. . . 185
8.6 The multi-vendor hypermedia application hierarchy of a production line drive unit. 187

Acknowlegements

I would like to thank my supervisor Professor Wendy Hall, for providing me with the opportunity to work within a lively and progressive research group. I have attended more than my allotted number of local and foreign conferences as a direct result of Wendy's support, and for that I am very grateful. Thanks also to Dr. Dave DeRoure for acting as my second supervisor and for providing the doughnuts at early brainstorming sessions. My thanks also extend to Dr. Hugh Davis for allowing me to bounce ideas off him on numerous occasions. I would also like to take this opportunity to thank Dr. Antoine Rizk for agreeing to be my external examiner.

Thanks are also due to all my friends in the Multimedia Research Group. To Jonathan Dale for the laughs, debates and beers. To Dr. Mark Dobie for his interest, being a LaTeX guru and hours of meticulous proof-reading. To Dr. Ian Heath, Mark Weal and Dr. Gary Hill for essential technical information on Microcosm. To Andy Lewis, Emilia Mendez and Gary Wills for general discussions on hypermedia. To Karen Bidwell and Julie Smith for guiding me through the bureaucracy. To everyone who posts to the Multimedia Research Group *jokes* mailing list for providing some light relief, although the quality at times is dubious.

I must thank my entire family for their love and encouragement. But most significantly, I remain indebted to my wife Sue for her friendship, love and support.

1 Introduction

Over the last three decades the proliferation of digital computers has led to an exponential growth in the volume of electronic information being generated. Hypermedia systems have recently evolved through our requirement to integrate, manage and understand our information better.

1.1 The Origins of Hypertext

Historical texts serve to demonstrate the importance of citing supplementary and contrasting works when placing an argument in its logical context. Authors often employ schemes such as nested commentaries and annotations to act as a corollary to a theme. Examples of modern works such as scientific journals and reference books also exhibit these traits, with clear labelling of any dependencies upon other sources.

Each work in this genre could be placed within a network of *nodes* from which pointers to other pertinent sources emanate. Primitive texts that embody these principles, such as the writings of Aristotle and the Talmud, are among the earliest examples of hypertext. In 1945 Vannevar Bush [35], a scientific advisor to President Roosevelt, predicted an information explosion and was the first to suggest that automation could assist with the storage and exchange of such association trails. Contemporary forms of hypertext are able to harness the processing power provided by modern computer systems to deliver such material.

1.2 Characterising Hypertext and Hypermedia

Hypertext is a flexible definition applied to the notion of non-linear text, the fundamental principle being that segments of text are designed to be read in any order, as decided by the reader depending on their needs and interests. Hypermedia is the term used to describe the extension of hypertext to include multimedia information.

Systems that provide hypermedia functionality are hypertext systems that are capable of presenting and linking over a variety of media.

The terms hypertext and hypermedia are among the recent additions to the vocabulary of the computer literate, with both terms generally accepted as synonymous with the technology. Although a plethora of definitions exist for hypermedia, arguably one of the better descriptions was presented by Theodore Nelson [144], responsible for the neologism.

> It should have been obvious to anyone that general interactive media would appear and proliferate. Text, graphics, audio and video can now come alive in unified, responding, explorable new works that present facts and ideas; hypermedia. Unlimited new forms of connection and branching now offer the chance to explore ideas - to follow different lines of thought, different forms of exposition, different connections in a subject, in ways never before possible. The sequential writings and media of the past have given us only the dimmest precedents.

1.3 Defining Hypermedia

A definition of hypermedia based upon the aims of the technology was presented by Halasz [97], who wrote:

> Hypermedia is the style of building systems for the creation, manipulation, presentation and representation of information in which: the information is stored in a collection of multi-media nodes; the nodes are explicitly or implicitly organised into one or more structures (commonly, a network of nodes connected by links); users can access information by navigating over or through the available information structures.

A fundamental requirement of any hypermedia system is to provide mechanisms for defining associations between interrelated pieces of information. This is normally achieved by selecting a region of interest, an *anchor*, in one document and connecting it to another document (or anchor within that document). These associations, commonly referred to as *links*, can then be traversed by activating the source anchor (typically with a mouse click) enabling non-sequential access to information.

1.4 Distributed Hypermedia

Users are becoming increasingly aware that access to networks, in particular the Internet, greatly enhances the amount of information at their disposal. The unmitigated success of the World Wide Web (WWW) bears testimony to the previously unsatisfied need for a system able to integrate and deliver distributed information. The profile of hypermedia has been significantly raised by the WWW, which has endorsed hypermedia as an appropriate technology for accessing and navigating information spaces.

The native architecture of the WWW is capable of providing basic hypertext functionality. Although the WWW is open in terms of the platforms and protocols supported, it is inherently reliant upon a specific markup language (HTML) for denoting embedded hypertext links. Over the last decade, the hypermedia research community has sought new models and designs for systems that are ever more open and flexible. The requirements for an open hypermedia system are examined throughout this thesis and a treatment of these issues is conducted with added emphasis on distribution. These requirements and their consequences provided the motivation for the successful implementation of Microcosm TNG, an open and extensible distributed hypermedia system.

1.5 Thesis Structure

Chapter 2 provides a personal view of the motivations and subsequent development of open hypermedia systems, hyperbases and link services. An account of the evolution of prevalent ideas, models and standards is presented followed by a distillation of the criteria for open hypermedia.

A broad survey including many of the influential hypermedia systems is presented in chapter 3. Each system is categorised according to the flag taxonomy and reviewed in terms of the openness of the design.

Chapter 4 contains a brief technical description of the open hypermedia system Microcosm, developed by the Multimedia Research Group at the University of Southampton. Subsequent research reported in this thesis builds upon the foundations established in the design of this system.

Chapter 5 provides an overview of the distributed information services currently available and continues with a review of distributed hypermedia systems.

Communication is central to distributed systems. In chapter 6 a comparison of the existing models of communication for Microcosm is presented after which a new distributed communication framework is described in some depth.

Microcosm: The Next Generation (TNG), a distributed open hypermedia system, is described in chapter 7. The original requirements for open hypermedia are revisited in this chapter but analysed with more of a focus on distribution. Three core requirements are distilled and provide the motivation for the new system. The remainder of the chapter describes the design of the system and the way in which users can share information, structure and link services through the notion of a *hypermedia application*.

The notion of a hypermedia application is explored further in chapter 8. The anatomy of a hypermedia application is considered in an attempt to improve modular construction and make it more amenable to both maintenance and reuse. The implications of extending the definition of a hypermedia application to include one or more existing hypermedia applications is explored.

Finally, chapter 9 describes some of the potential areas for future research that have arisen from the work reported in this thesis. A number of conclusions from the research undertaken are then presented.

1.6 Declaration

Although the entire thesis is the author's personal view of the field, the original contributions are described in chapters 6, 7 and 8. The work in this thesis has been conducted within a collaborative research group. It is all the author's own work, with the exception of that described in sections 6.7, 7.10.3, where other members of the research group have in some way contributed.

2 The Importance of Being "Open"

2.1 The Open Systems Movement

Back in the 1960's and 1970's customers became aware that they were locked into their IT suppliers through investment in proprietary hardware and software products. Around the same time the UNIX operating system became popular. The advent of UNIX did much to relieve this situation of "vendor lock-in" as it could be obtained inexpensively, ran on a variety of platforms and provided common functionality across them. UNIX provides a way for systems to become more open in terms of reducing dependence on one vendor. It is for this reason that it is closely associated with the world of open systems.

As many vendors recognised that their customers were no longer obliged to buy from a single vendor, attitudes in the industry began to change. Even the largest companies became aware that they could no longer afford to ignore this phenomenon. As a result, the computer industry reluctantly released its grip on customers and has been coerced into adopting a more open philosophy.

The open systems philosophy has had, and continues to have, a profound and pervasive effect throughout computing. We have witnessed contributions towards this goal from many computing disciplines, not least from that of the hypermedia community. In this chapter the meaning of openness in the context of hypermedia is examined and the benefits of designing more open architectures are explored. The research effort directed toward this end is charted, noting the significant milestones, and concluded by distilling the requirements for an open hypermedia system.

2.2 The Tide of Change

As the cost of computing power decreased and the appropriate technology became available, the early visions of Bush became attainable. The first hypermedia systems to appear indicated that material linked by association could provide another useful mechanism for information access and management, alongside that of database systems and information retrieval engines. The design and features of these early systems are well documented [43].

The first hypermedia systems were generally monolithic applications that provided an entire environment through which *hyper-linked* static reference material could be viewed. Once documents had been imported into these hypermedia systems they were no longer readable by the original applications that had been used to create them. Many of these hypermedia systems converted each new document into a proprietary format before it could be processed, e.g. Guide [151] and KMS [1].

Almost without exception these early systems employed a scheme whereby the positions of link anchors were denoted by symbolic representations stored directly within the documents to be used. This method for specifying links is referred to as *embedded markup*. Unfortunately this approach makes it very difficult to modify or extend the basic linking model. Although some systems could be extended to support additional media, this was via the invocation of stand-alone applications. This meant that any subsequent extension could not function as a fully integrated component of the hypermedia system.

It has been recognised that as users become increasingly familiar with a new program or computer system, many are soon able to use the software to its full capacity while also identifying novel uses that the designer(s) may not have envisioned. The lack of extensibility within these closed monolithic systems stifled and precluded users from experimenting in this way.

The combination of these characteristics led to these hypermedia systems existing in a vacuum, isolated from the ever more integrated host environment, hence examples of such systems were referred to as *closed*.

The international hypertext conference in 1987 was the venue at which many of these concerns were first voiced. The presentations by Andries Van Dam and Frank Halasz, both of which are now widely regarded as seminal, alluded to these problems and provided the community with a research agenda. This event proved to be the catalyst for the research effort that ensued.

In his keynote address, Van Dam [177] recounted his involvement with hypertext at Brown University through the development of the Hypertext Editing System (HES), the File Retrieval and Editing System (FRESS) and the establishment of IRIS[1]. He then proffered his view of the crucial areas that the community should focus upon. His second point described the limitations of closed hypermedia systems within the desktop environment and that greater integration could be realised by offering hypermedia functionality as an operating system service.

> ...Ted [Nelson] talks a lot about the docuverse, a mythical entity out there that is all inclusive and contains everything. But instead, right now we are building docu-islands; none of our systems talk to each other, they are wholly incompatible. So we are all working on the same agenda, more or less, but we can't exchange stuff; there is no exchange format, there is no universality, and furthermore, our systems are *closed* systems...
>
> ...So it is not enough to be able to bundle the HyperCard package with every Mac you buy. It really ought to be migrated down, become part of the toolbox, so that application programmers can take their applications and take advantage of a standard linking protocol that works within and between applications.

He then stressed the importance of standards as a way to achieve this end, but acknowledged that hypermedia, as a research topic, is immature and that attempts to standardise too soon could compromise experimentation. He continued by emphasising that all users are different and that future systems should be versatile enough to accommodate varied configurations, allowing each user to customise individual aspects of the system to their requirements.

> But we don't want to put things together in a way that there is only one point of view, because if we have learnt one thing from interactive tools up to now, it is that multiview is the way people work. You cannot have it just one way.

The paper presented by Frank Halasz drew heavily upon his experience and involvement with Notecards [98], a hypermedia system produced at Xerox PARC. He

[1] Institute for Research in Information and Scholarship

identified what he considered "Seven Issues for the Next Generation of Hypermedia Systems" [97]. These seven issues had implications for almost every aspect of hypermedia system design. He advocated novel search and query mechanisms to complement existing hypermedia navigation methods. Support for the management of dynamic information was considered a key feature if hypermedia systems were to play an integral role in the real world. This requirement implied the need for versioning that could support auditing processes. He also considered hypermedia to be an ideal technology for supporting collaborative working practices. Finally, Halasz noted that hypermedia is a generic technology allowing it to be useful across many task domains, but to offset the difficulties caused by their generic nature hypermedia systems should be extensible and tailorable. He recommended that future hypermedia systems should provide simple mechanisms for casual users to make small changes and experienced programmers to effect significant changes.

Brown [29] muses over the reasons why many innovative tools do not succeed outside the research laboratory. He concurs with Van Dam by observing that greater integration is key to the future of hypermedia systems:

> A second reason for failure may be that a [hypermedia] tool is an island to itself and cannot be combined with other tools.

He goes on to say:

> We must also work with existing tools such as spelling checkers or encryption programs...we must fit the world as it is rather than the world as we would like it to be.

In essence Brown suggests that evolution, and not revolution, is the way forward for the next generation of systems.

Meyrowitz [140] argues that the reason for the failure of early hypermedia systems was that they were:

> virtually all insular, monolithic packages that demanded the user disown his or her present computing environment to use the functions of the hypermedia system.

He believes that link creation and management operations should be available to the users with the same ubiquity as are the cut, copy and paste operations in modern desktop windowing environments. He reiterates the call made by Van Dam, for linking functionality to be incorporated within standard computing toolboxes to enable significant advances in application integration to be made.

> Only when the paradigm is positioned as an integrating factor for all third-party applications, and not as a special attribute of a limited few, will knowledge workers accept and integrate hypermedia into their daily work process.

These authors recognised the importance and benefits of greater integration with the desktop environment, together with facilities for extending and customising the core system. These opinions converge with those of Bush, Englebart and Nelson, who foresaw an integrated environment for the management and sharing of information.

2.3 The Dexter Reference Model

The research community had recognised the need for a reference model for hypermedia, and workshops were organised with a view to producing one. These workshops, held in 1989, were attended by representatives from the major US research groups and a variety of models were proposed. The *standard* that was accepted by the delegates was the Dexter Reference Model [99, 100], but frequently abbreviated to Dexter. The original publication [99] provides a very formal description of the Dexter model using the Z specification language [106], but the later publication [100] introduces the model to non-mathematicians in a more accessible manner.

The Dexter Model attempted to provide a standard hypermedia terminology coupled with a formal model of the common abstractions found within contemporary hypermedia systems. At that time the most prominent hypermedia systems were Intermedia [189], Notecards [98], KMS [1], Neptune [57], Augment [64], Hyperties [167] and the Sun Link Service [154]. The authors hoped that Dexter could serve as a standard against which hypermedia systems could be compared with respect to their characteristics and functionality. It was also thought that the model provided a principled basis on which standards for interchange and interoperability between hypermedia systems could be developed. The Dexter Model divides hypermedia systems into three separate layers, as illustrated in figure 2.1.

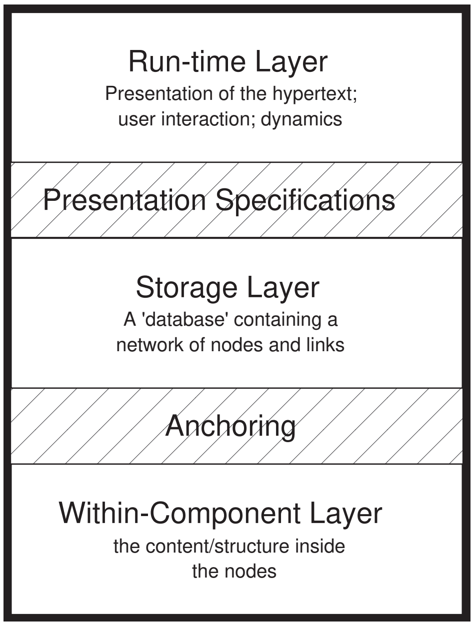


Figure 2.1: The layers of the Dexter Model.

The *runtime layer* provides a model for the presentation mechanisms for the user to access, view and manipulate the hypermedia network structure. This layer is intended to cover the dynamic, interactional aspects of hypermedia systems, without attempting to cover the details of user interaction with the hypermedia network.

The *storage layer* is where the focus of the model was concentrated, as it relates to the mechanisms by which link and node components are associated to form hypermedia networks. Links are stored as lists of link specifiers, with each link specifier containing a component identifier, an anchor identifier, a link direction and a presentation specification.

The *within-component layer* is concerned with the content of a component, node or document. The authors concede that it is beyond the remit of Dexter to attempt to model the structure or content of a component, leaving this layer largely unspecified. This can be performed by a database or the file system, but it is envisaged that structured document mechanisms will fulfil this duty.

Sandwiched between the runtime layer and the storage layer is an interface responsible for managing presentation specifications. Presentation specifications are details encoded within the storage layer that state how a component should be presented to the user. The advantage of this is that the way in which a component is displayed to the user can be a function of the hypermedia system, a property of the component itself and also the path taken to reach the component.

Between the storage layer and the within-component layer is a second interface that provides access to components through anchors. Anchoring is a mechanism used in Dexter to allow span-to-span links to be supported while maintaining a separation between the storage and within-component layers.

Although the Dexter Reference Model is considered by many to be the most important milestone in the recent history of hypermedia research, several authors have noted shortcomings, while others have found it necessary to extend the model. Grønbæk *et al*, whose DHM hypermedia system is arguably the most Dexter compliant system to date, extended the Dexter model to support the requirements for cooperative working [88, 89]. The notion of composites within Dexter has proved useful, encouraging a number of authors to propose extensions for a wider variety of structuring mechanisms to be supported [86, 69]. Suggestions have also been made about how to extend the Dexter model to manage temporal event specification [104], a necessary consideration for systems with sophisticated multimedia support. A discussion as to the suitability of Dexter as an interchange standard is postponed until section 2.7.

Grønbæk *et al* claim that even though the Dexter model pre-dates the majority of contemporary open hypermedia systems much of the terminology remains valid. Recent work by the DHM team [92] has demonstrated that it is possible to model such diverse systems as the World Wide Web and Microcosm with an extended Dexter model.

Although Leggett *et al* [126] suggest that the NIST standardisation workshops occurred prematurely, it is beyond doubt that the Dexter model has made a significant contribution to the design of hypermedia systems.

2.4 Impetus from Industry

Another seminal paper that contributed to the debate surrounding open hypermedia was published by Malcolm *et al* [135], who at that time worked for Boeing. They provide an insight into the way large corporations operate and the associated complexities in supporting such an infrastructure. The authors describe how as interdisciplinary teams are given the responsibility for designing new products and their life cycles, these teams would greatly benefit from a technology that could assist in maintaining a seamless thread from preliminary design through to manufacturing, training and customer support. They question whether hypermedia can help fill the current void.

Engineering data is generated using specialist tools in addition to office applications such as word processors and spreadsheets. The authors would like to see hypermedia playing the role of integrator:

> Hypermedia technology can be used to provide access to data *and* to manage the applications used to create the data. Although hypermedia is often thought of as a technology to deliver information, its use can be greatly expanded if it is perceived as an integrating technology.

This insightful observation was clearly demonstrated through the depiction of a workplace scenario in which members of a team collaborated to complete various tasks with the support of a fictitious *industrial-strength* hypermedia system. The following comment summarises how the authors perceived the state of affairs at that time.

> The hypermedia world is a chaotic swirl of conflicting protocols, interfaces, representations and capabilities, and in our assessment, today's tools are not mature enough to handle the range of user and computing environments in large aerospace companies. Though there are a few multi-user hypermedia tools with limited collaborative authoring capabilities, most are closed systems that cannot be integrated with the universe of applications and platforms used in the industry.

Although the hypermedia systems available commercially are presently inadequate to support an international enterprise, the authors believe that hypermedia shows much promise for the future. Before the next generation of hypermedia systems can qualify as *industrial-strength*, they consider that the following issues must be addressed satisfactorily.

Interoperability the ability to access and link across arbitrary platforms, applications and data sources. For this to be complied with in full requires multimedia data interchange standards and a link service available from within all applications.

Link Issues it is probable that users will want to attach additional information to links, for example, who created the link, the time and date it was created. Such attributes may suggest whether a particular link would be fruitful to follow.

User defined link types, although just another attribute, can be used to express an argumentation or reasoning process, such as a design rationale. In an engineering situation it is also conceivable that a link may initiate and control external processes.

In a collaborative setting, each user must be able to maintain private and public links. This will enable teams to share information while also allowing an individual the freedom to explore new ideas privately.

Templates provide a means of replicating common structures found within a hypermedia network. This allows information with a well defined structure such as reference manuals and training material to be easily incorporated and linked with the system.

Navigational Aids such as network browsing, query mechanisms, tables of contents, bookmarks and guided tours are required to help users locate information rapidly.

Configuration Control provides a mechanism whereby the respective versions of objects, links and hypermedia webs can managed. A scheme for applying permission rights and/or passwords to all entities should also be available. By tracking the evolution of these entities the system can help support commercial processes.

Concurrency Control exists to monitor, and prevent, any undesirable conflicts that may occur in a multi-user collaborative environment. The challenge is to provide this protection but at a fine granularity so as not to inhibit collaborative design practices.

Programmability provides the opportunity for extending and customising the functionality of the hypermedia system.

Operating Systems, Storage and Networks are a fundamental concern when considering the transfer of high quality multimedia information across widely distributed, heterogeneous networks of computers. This involves employing techniques for object replication, caching, compression and decompression.

It is clear from the issues raised by Malcolm *et al* that closed hypermedia systems will not suffice, and that they are looking toward the development of large-scale open hypermedia systems and standards to meet this need.

The closing plenary at the international hypertext conference in 1991 was the occasion chosen by Frank Halasz of Xerox PARC to revisit the seven issues he placed before the research community in 1987. He began by reviewing the progress made by the community as a whole with respect to his agenda. He welcomed the contributions to search and querying [62], addition of composites [99, 168, 163], use of virtual structures [22] and computation over hypermedia [142, 22]. Even though published evidence of progress in the areas of versioning and support for collaborative working within hypermedia systems was hard to find, Halasz remained convinced as to the importance of these areas. The final issue on his agenda was that of tailorability and extensibility. He noted advances on this front using scripting languages [142], schemas [136, 72] and templates [40]. He predicted that open hypermedia systems, composed of interacting processes with individual responsibilities, will change the way in which tailorability and extensibility are provided.

Halasz acknowledged the contribution by Malcolm *et al* [135] and stated that a user centred technology such as hypermedia needs real users to assist in directing the research. After revisiting the original seven issues, Halasz presented seven new issues fashioned and influenced by recent hypermedia research and progress in technology.

Ending the Tyranny of the Link Halasz cites SuperBook [62] as an example of a hypermedia system in which information retrieval is used as the primary means of traversing the information space, as opposed to activating links in the traditional model. In essence he recommends relaxing the constraints on the definition of hypermedia so as to include other browsing and navigation tools such as information retrieval, relevance feedback and expert systems.

Open Systems Reflecting upon the design of hypermedia systems, Halasz had this to say:

> It's very clear that the hypermedia system of the 1980's, the monolithic hypermedia system, things like Notecards and even Intermedia, is no longer viable. They're going to be replaced by open hypermedia systems in which you have independent but communicating components which, taken together, produce hypermedia functionality.

The question remains as to the most efficient and appropriate decomposition of the hypermedia functionality in these components. He prompted the research community to investigate inter-application communication facilities, such as OLE [65] and OpenDoc [9, 10], as a means of assembling these components.

User Interfaces for Large Information Spaces User interface designers have been working for some time to develop ways in which large volumes of information can be visualised conveniently on the screen of a workstation. Partial solutions have been published, for example Fisheye views [67] and 3D models [37]. As the volume of digital information increases, Halasz foresaw a burgeoning need for such tools.

Very Large Hypertexts It has been suggested by various proponents of hypermedia that this technology is suitable for managing vast repositories of information. Halasz argued that such a claim is unproven, as at that time researchers had only experimented with systems managing several thousand documents. International enterprises will require industrial strength systems with scalable solutions to document management. Related issues include the need for suitable interfaces, discussed above, and the diversity of document formats that exist within distributed computing environments.

Between them Halasz and Malcolm *et al* outlined a number of non-trivial issues which in 1991 the research community needed to consider in the design of future generations of hypermedia systems. Many of these remain research issues today.

2.5 Various Approaches to Delivering Open Hypermedia

Ever since Van Dam and Halasz hinted at the benefits of adopting a more open approach to hypermedia, the research community have been working toward this target. This section attempts to cover briefly the various approaches explored.

2.5.1 Scripting Languages

As alluded to earlier, many of the monolithic generation of hypermedia systems provide scripting languages which, to varying degrees, allow users to extend functionality and customise generic behaviour. This approach can be found in systems such as HyperCard (HyperTalk language) [8], Guide (Logiix language) [30] and KMS (Action language) [1].

Halasz [97] questioned whether scripting languages provide an accessible solution to the issue of extensibility. He recommended a dual approach; a simple mechanism which makes it easy for the majority of non-programming users to make small changes to the system with minimal effort, coupled with a sophisticated mechanism that allows expert programmers to effect major changes. Multicard [159] and HyperDisco [183], both reviewed in chapter 3, fulfil this criteria but also allow scripts to be attached to individual components and anchors to extend behaviour.

2.5.2 The Toolkit Approach

Another approach explored by various groups was to supply a library with a published API[2], or *toolkit*, to provide improved access to the functionality of the hypermedia engine.

Perhaps the first toolkit to appear was the Hypertext Abstract Machine (HAM) [36]. This toolkit provides many of the classical hypermedia features and storage structures. A more generic offering was suggested by the authors of the Eggs/HOT

[2] Applications Programming Interface

toolkit [155]. The internals of the Notecards system were also revealed via an API [175], allowing applications written in Lisp dialects to be easily integrated with the system.

A common characteristic shared by the systems described above is that they require each application wishing to use the hypermedia services to be tightly linked to the toolkit.

2.5.3 Link Services

A natural progression from the toolkit approach was to define a communications protocol so as to achieve a loose coupling between the hypermedia systems and any application that respected the protocol. The first published example of such an approach was the Sun Link Service [154], the advantages of which are summarised by the author below.

> An open system implies that the system can be extended to mediate different heterogeneous implementations; integration implies much greater cooperation and conformity. The Link Service modulates this tension by defining as simple and unrestrictive a protocol as possible. In this way we preserve the autonomy of individual tools, yet provide some measure of integration.

As autonomy is preserved, a variety of interaction styles can be accommodated because the link service does not prescribe the way in which the information is presented to the user. The link service can be viewed as an application positioned above the storage layer that runs continually and unobtrusively. It assumes responsibility for handling communications with all applications wishing to use the hypermedia services. Davis [48] defines a link service as *lightweight* if the participating applications may elect to use only a subset of the full functionality.

Typical examples of hypermedia systems providing a link service layer include Sun Link Service [154], Microcosm [66, 103], Intermedia [96], PROXHY [119], Multicard [159] and Chimera [6], all of which are reviewed in chapter 3.

2.5.4 Hyperbases

In 1991 Halasz and Malcolm *et al* raised doubts as to whether current hypermedia technology could scale adequately to support the size of information repositories such as those envisaged for digital libraries of the future. For this reason, the NSF[3] sponsored a workshop to promote discussion on this issue [127]. A *hyperbase* is merely a hypermedia database, but the following definition makes clear the responsibilities of this entity.

> Hyperbase systems must support not only storage and manipulation of complex multimedia data but also storage and manipulation of connectivity data. In addition to the usual requirements for permanence of data, controlled sharing, backup, and recovery, hyperbase systems must be capable of modelling complex interrelationships and providing direct support for novel data types. Hyperbase systems must also be able to handle lengthy transactions, transactions unique to hypermedia, massive distribution of functionality, extensibility, notification, versioning of hypermedia data and structure and scaling to the hundreds of terabytes.

The workshop participants agreed that a hypermedia system has a three layered architecture. The application layer comprises familiar desktop tools such as editors, spreadsheets and mail readers. The hyperbase layer provides a hypermedia data model to the applications layer and interacts with the storage layer to save the multimedia data, contexts, nodes, links, anchors, etc. Typical examples of hyperbase systems include HB2 [126], CHS [163], DGS [165], DHM [88] and HyperDisco [183].

The definition of a hyperbase, introduced earlier, asserts that all documents are stored within the hyperbase. This requirement makes it difficult for third party applications because access to documents is usually through the file system. Several of the modern hyperbases now permit documents to reside either on the file system or within the hyperbase and also act as a link service.

[3] National Science Foundation of the USA

2.6 Hypermedia Standards

Randall Trigg, during his closing keynote address at the international hypertext conference in 1996, reaffirmed the widely held belief that open hypermedia can play a pivotal role as an integrating technology. Although some progress has been made on this front, many commentators argue that before seamless integration can occur widespread agreement upon standards for application interoperability and information interchange must be reached.

Greater application interoperability can be realised by creating systems that use and conform to published communication protocols. The merits of such a protocol can be judged by its completeness, degree of acceptance and use. Similarly, the interchange of information can be greatly eased by adhering to recognised standard formats. The expense incurred by converting to another format can be considerable, consequently users must be convinced of long term stability in addition to the completeness of the format.

There are two types of standards: *de facto* standards and *de jure* standards. *De facto* standards are those which are commonly used, usually emanating from a single vendor. The advantage is that they tend towards the latest technology. Conversely, *de jure* standards are the result of a formal committee which, following numerous revisions, eventually promulgate standards. Although *de jure* standards are vendor-independent, they inevitably reflect older technology. The advantage is that best practice across the entire industry is considered and as a result a more complete standard can be achieved. The standardisation effort in the field of hypermedia is reviewed below.

2.6.1 HyTime

The Hypermedia/Time-based Structuring Language [146, 53] was granted the status of international standard in 1992 [114]. It was developed during the era of monolithic hypermedia systems and was, in part, conceived as a solution to the problem of information interchange between different systems. The HyTime standard evolved from previous work upon the specification of a music description language (SMDL) [145]. As would be expected from a derivative of a music specification language, HyTime has strong support for the scheduling and interconnection between the related components of a hypermedia document.

HyTime is not an application, nor a document architecture, but a mechanism for expressing document architectures in SGML[4]. It is often described as a *meta-DTD*[5] since it provides a set of standard architectural forms which can be used to construct document architectures. The standard defines an extensive meta-language for hypermedia documents, which include general representations for links and anchors, a versatile framework for positioning and projecting arbitrary objects in time and space, and a structured document query language (HyQ).

Relatively few implementations of subsets of HyTime exist, partly due to the complexity of the standard. HTML 2.0, discussed in chapter 2.6.3, has recently been defined as an SGML DTD. It has been suggested [33] that once SGML is properly supported within World Wide Web browsers, support for HyTime is technically feasible and would provide an improved hypermedia framework for the Web.

2.6.2 MHEG

The collaborative effort by MHEG[6] has also been to develop an international standard for the exchange of hypermedia objects between applications [19]. The MHEG standard, called *Coded Representation of Multimedia and Hypermedia Information Objects*, is a container architecture which allows media objects (initially text, graphics, audio, still and moving pictures) to be coupled with instructions for their presentation and behaviour.

MHEG differs from HyTime in that it is intended for hypermedia applications requiring real-time information interchange and addresses the problem of exchanging hypermedia objects explicitly for presentation. Encoded along with the multimedia objects are the hyperlinks, display and synchronisation semantics. These objects can be encoded using either ASN.1[7] or SGML. Evidence exists to show that a MHEG engine is significantly simpler than a HyTime engine and consequently demands less

[4] Standard Generalised Markup Language

[5] Document Type Definition, a part of SGML

[6] Multimedia and Hypermedia information coding Experts Group

[7] Abstract Syntax Notation

resources [33].

2.6.3 HTML

HTML (Hypertext Markup Language) [17] is the proprietary document format of the World Wide Web for indicating to browsers the presence of embedded link anchors and specific content decoration. As mentioned in section 2.6.1, HTML version 2.0 has been accepted as a standard. Like HyTime, HTML is also based upon SGML with a DTD available for version 2.0 of the standard.

There are a number of disadvantages with HTML: the use of embedded link anchors within HTML means that no distinction between content and structure is made; it has a fixed data model supporting only point-to-point links; as Van Dam [177] noted, one DTD, and hence one view of the information, is not sufficient to serve all users equally well. Although it is easy to criticise HTML and the World Wide Web, its success may well be a testimony to its simplicity and to the speed at which it evolves.

2.6.4 HTTP

HTTP (Hypertext Transfer Protocol) [15] is the simple protocol spoken between World Wide Web clients and servers. It has been accepted as an IETF[8] standard protocol with subsequent versions having been ratified. The protocol specifies a minimal number of operations, each taking a Uniform Resource Locator (URL) and a protocol version number.

2.7 Hypermedia Interchange

According to Leggett *et al* [126], the sharing of hypermedia among distributed collaborators can be accomplished through two mechanisms:

exchanging a network of information involves the hypermedia being passed from collaborator to collaborator via a channel such as electronic mail. If the collaborators are using different hypermedia systems, then the hypermedia must first be translated to a format that can be interpreted by the receiving system.

[8] Internet Engineering Task Force that promulgates standard protocols for the Internet

accessing a shared network of information refers to the situation where each of the collaborators can browse the hypermedia using one or more applications that support an open protocol. They refer to this scenario as *hypermedia-in-the-large*.

The first mechanism described above is that of hypermedia interchange. Many hypermedia systems exist, with few sharing an identical underlying model of the data. By defining an intermediate format, theoretically it is then only necessary for each hypermedia system to be able to convert to and from this format to achieve this goal.

Leggett *et al* [125] conducted a series of experiments to establish whether the Dexter Model is a suitable candidate for an intermediate exchange format. Two hypermedia systems were selected, KMS [1] and Intermedia [189], that had radically different underlying models and an export facility [157]. Although the modelling capabilities of Dexter proved adequate, the authors revealed a number of difficulties with this approach. For example, translating the source format to the intermediate format and finally to the destination format, resulted in many of the source system's features being remapped to comply with the model of the destination system. Unless measures are taken, such as special attributes set within each component prior to export, when the reverse process occurs the hypermedia would be incorrectly remapped to the original source system. This loss of structure violates the condition that systems should import and export the identical interchange format. The features of the respective data models that raised these problems were one-to-many links, link directionality and the representation of link webs or contexts.

As explained in chapter 2.6.1, HyTime was intended to promote hypermedia information interchange and as such similar research has been conducted to examine its completeness. Less comprehensive examinations have shown that both Microcosm [38] and the Amsterdam Hypermedia Model [105] can also be modelled using HyTime.

2.8 Distilling the Requirements for "Open" Hypermedia

Since 1987, there has been an irrevocable move towards increasingly open hypermedia systems, evidenced by the research outlined above. Interest in this specific area of hypermedia research inspired a succession of workshops which were staged in Konstanz [11], Edinburgh [185] and Washington D.C. [180]. Various attempts have been made by individuals and workshops to define the term *open hypermedia*, so what does it actually mean ?

2.8.1 A Definition

As with any active field of research it can prove difficult to reach a conclusive definition, but the following recent six point definition of open hypermedia, supplied by Davis [48], is considered to embody the consensus of opinion of the vanguard.

An open hypermedia system is a hypermedia system where the term open implies the possibility of importing new objects into the system. A truly open hypermedia system should be open with regard to:

1. *Size:*

   It should be possible to import new nodes, links, anchors and other hypermedia objects without any limitation to the size of the objects or to the maximum number of such objects that the system may contain, being imposed by the hypermedia system.

   If open hypermedia systems of the future are to support application domains such as digital libraries and industry, it is important that any design decisions will not prohibit them from scaling up to meet these needs. As both the size and number of objects rises inexorably external and distributed storage will become necessary. For each object to be known to the system it must be uniquely addressable, and schemes for addressing large distributed information repositories must be flexible and scalable.

   Malcolm *et al* reminds us that information in these environments is dynamic, regularly being revised and modified. It is crucial therefore that new information can be seamlessly integrated within an expanding resource base.

2. *Data formats:*

   The system should allow the import and use of any data format, including temporal media.

   It is impossible to provide intrinsic support for all future, or even archaic, formats. At worst, the system must be able to invoke a third party application with which to present the media. The ideal solution is to make use of the original editor with which the information was created, but provide access to the link service, which relates to the next point.

The introduction of temporal data (such as video and audio) and virtual data (dynamically computed) poses numerous non-trivial challenges, such as how anchors are specified and presented in a consistent and flexible manner.

3. *Applications:*

   The system should allow any applications to access the link service in order to participate in the hypermedia functionality.

   As hinted at in the previous point, one particular goal is to provide easy access to an open link service from within every desktop application. This would enable the original editor that was used to generate the document to also be used by the link service to present the document. This approach has a number of attractions; it is likely that the user is well acquainted with the editor; it is not only pointless, but also very expensive to duplicate effort in developing a specific media viewer. A variety of techniques for achieving closer integration with third party applications have been published [119, 159, 52, 6, 183, 179].

   If, and when, an open link service is integrated with the desktop it is hoped that users will perceive hypermedia as an enabling technology rather than a separate application as is currently more usual.

4. *Data models:*

   The hypermedia system should not impose a single view of what constitutes a hypermedia data model, but should be configurable and extensible so that new hypermedia data models may be incorporated. It should thus be able to interoperate with external hypermedia systems, and to exchange data with external systems.

   While Dexter provides a general model for hypermedia, it cannot hope to be sufficiently versatile to model every situation. This point states that an open hypermedia system should not limit users to a single data model, such as Dexter, as it may not be appropriate. A number of alternative data models for hypermedia have been proposed, including Trellis [68], Sets [153] and Dexter [99]. Hyperform [181] is an example of an open hypermedia environment with a data model that can be dynamically customised. New instances of hypermedia objects can be generated at runtime through the use of Scheme scripts.

The need for an extensible link engine able to support multiple data models is clear, but this requirement exacerbates the problem of hypermedia interchange. This issue was introduced in section 2.7, where some of the problems of mapping between hypermedia data models were explained. If the number and diversity of data models increases the problem of mapping between them will become more difficult.

5. *Platforms:*

   It should be possible to implement the system on multiple and distributed platforms.

   An ideal open system should be able to operate across a variety of platforms, but supporting this ideal requires very portable source code. By conforming to a core set of standards, using only a common set of services and by observing published protocols, a high degree of heterogeneity can be achieved. It is inevitable that a proportion of platform dependent idiosyncrasies will deny the routine porting of systems, but these can be kept to a minimum.

   Users are becoming increasingly aware that access to networks, in particular the Internet, greatly enhances the amount of information at their disposal. Hypermedia can provide an organised, unified view of physically distributed information. As greater demands are placed upon hypermedia systems to support more computationally intensive link resolution and querying it is necessary to investigate distributed architectures as a way of providing more flexible and scalable solutions. This aspect is the focus of this thesis.

6. *Users:*

   The system must support multiple users, and allow each user to maintain their own private view of the objects in the system.

   The ability to share information is one of the central benefits of a modern open hypermedia system with support for multiple users. Malcolm *et al* [135] called for private and public workspaces, but this can only be supported if all hypermedia link information is held in external databases. This then allows the appropriate public, or private, sets of links to be applied to the information.

Preserving document sanctity, by storing the links separately, has a number of interesting advantages. Each individual can create multiple perspectives on the information. For example, a collection of historical documents could be navigated from either a chronological or geographical perspective depending upon the web of links applied. As all documents remain untouched by links, it is also possible to create links to and from read only media (such as CD-ROM) and to documents for which the user does not have write permission. All users can now be promoted to the status of author or publisher [144, 143]. Bush [35] was the first to recognise a collection of artefacts augmented by a set of associative links and personal annotations as a valuable commodity.

2.8.2 A Conceptual Model

The above definition provides an exhaustive list of requirements for open hypermedia systems. An alternative perspective on open hypermedia is provided by the Flag Taxonomy [150], which attempts to capture the functionality and interaction of hypermedia systems in such a manner as to aid classification.

Although some dissent exists in the research community as to whether Dexter is the appropriate reference model for hypermedia, it has provided a principled basis upon which these discussions can take place. The authors of the Flag Taxonomy sought to provide a parallel reference model for open hypermedia systems.

The Flag Taxonomy is a conceptual model[9] which divides a hypermedia system into four functional modules. Each module provides functionality that can be used by its two neighbouring modules through the available protocols. Two important distinctions are evident from the Flag. The horizontal axis highlights the separation between content and structure, while the vertical axis highlights the separation between storage and runtime. This can be seen in figure 2.2. By assigning the semantics of real hypermedia systems to the four functional modules, the Flag Taxonomy provides a succinct, but simplistic way of comparing and contrasting them.

[9] Although based upon the Danish flag, there was some controversy as to whether a Union Jack would have proved more suitable!

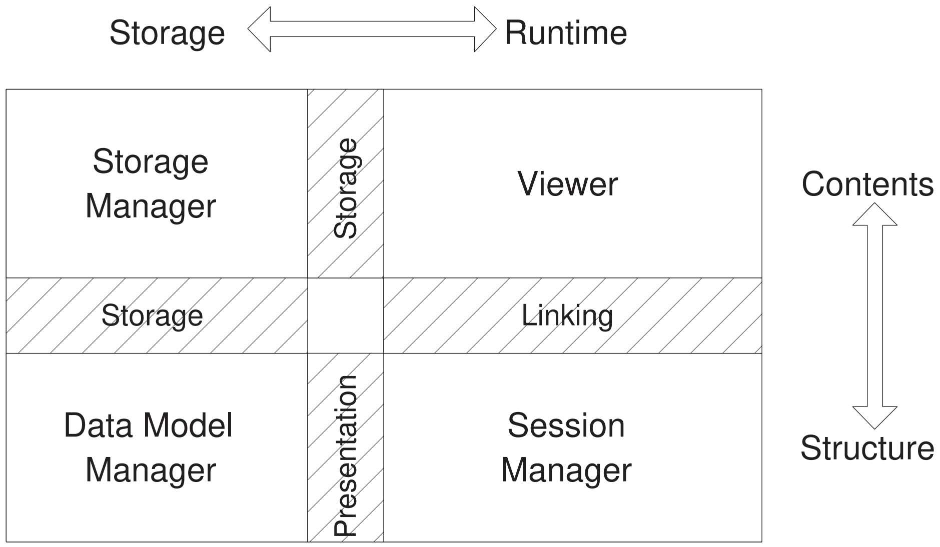


Figure 2.2: The Flag Taxonomy.

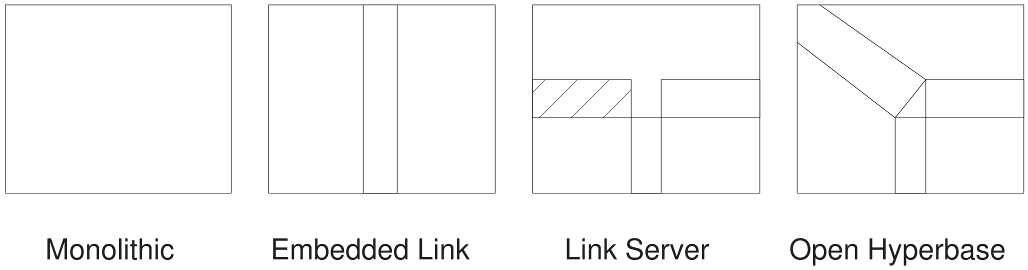


Figure 2.3: The broad categories distinguishable using the Flag Taxonomy.

Existing hypermedia systems can be classified into broad categories by cutting the Flag in various ways, seen in figure 2.3. A white box denotes a functional module, close parallel lines denotes a protocol and a greyed protocol indicates that it is unused.

A software developer may consider a hypermedia system with a single well-defined protocol to be open, but according to the Flag Taxonomy a linking protocol should also exist between the session manager and the viewer. This reduces the categories of hypermedia system that can be considered open to that of the link server and open hyperbase, as these do not impose a particular data model upon the viewer.

While the Flag Taxonomy provides a convenient metaphor for classifying hypermedia systems on a basic level, it lacks the detail required to distinguish between various systems of the same genre. When compared against the six point definition provided earlier, it is clear that the Flag Taxonomy fails to address the issues of data models, platforms and users.

2.8.3 A Protocol

Work is in progress to establish and promulgate an open hypermedia link protocol (OHP) [51] such as that called for by Van Dam, Meyrowitz and many more since. The initial draft of the proposal attempts to define a protocol through which applications such as word processors, graphics viewers and drawing programs can communicate with a link service to provide access to a range of hypermedia services.

The authors consider that the field has matured sufficiently to enable common themes to be abstracted from the research into open hypermedia systems. The major influences upon OHP are the Multicard M2000 protocol [159] and the Microcosm message model [66]. The authors acknowledge that this is a first attempt and OHP will undoubtedly undergo numerous revisions, but it is a positive and interesting development.

2.9 Are There Disadvantages to Openness ?

The benefits of open hypermedia have been presented throughout this chapter, but, as one might expect, there are some drawbacks that accompany this approach. Maintaining data integrity becomes a serious challenge once the decision is made to store link information separately from the documents. The editing and versioning problems are well understood, and the reader is referred to Davis [48] for a detailed exposition.

2.9.1 The Editing Problem

The editing problem occurs when the contents of a document are modified to the extent where the anchors, held in an external database, no longer correspond to the intended locations. Systems which use embedded markup to denote anchor locations do not tend to suffer to the same degree, as when the content moves so too do the anchors.

2.9.2 The Versioning Problem

If a new version of a document is recorded following each modification it is possible to keep track of all of the changes throughout its lifetime. In most situations this allows the hypermedia system to maintain the integrity of the anchors. When a user wishes to view this document the question arises as to which version is appropriate to display.

2.10 Summary

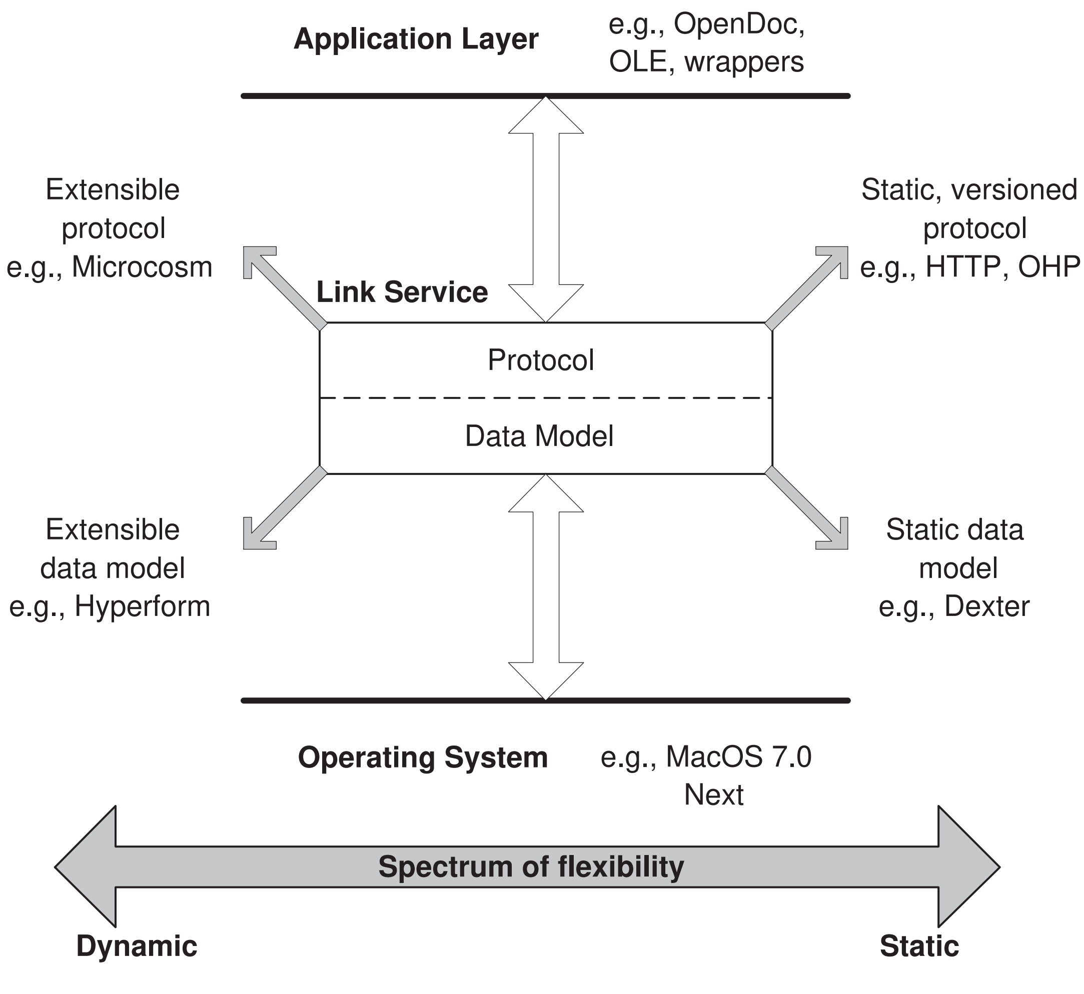


Figure 2.4: Summary of the directions of research into open hypermedia.

In this chapter a brief historical account of the need for and the subsequent development of experimental open hypermedia systems has been presented. The chapter concludes with a definition and a model of what the research community believe constitutes an open hypermedia system.

Much progress has been made toward the ideal of open hypermedia and figure 2.4 attempts to illustrate a number of dimensions in which openness is being researched and increasingly stretched. A variety of techniques have been published for greater application integration [168, 159, 50, 90, 6, 52, 183, 179]. Novel approaches advocating closer integration with the operating system are also under investigation [129, 148]. For the link service both extensible communication models [50] and versioned, standard communication protocols [15, 51] have been suggested. Similarly, for the link engine both extensible data models [181, 183] and revisions of standard data models [90, 69, 104, 92] have been published.

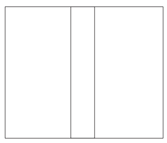
The overriding aim of the above research is to enable hypermedia to become a ubiquitous service and not simply a publishing tool for static information. The next chapter reviews a number of influential hypermedia systems which address many of the issues introduced in this chapter.

3 Taxonomic Review of Open Hypermedia Systems

3.1 Introduction

From the late 1980's it became clear that "closed" monolithic hypermedia systems were neither flexible nor extensible enough to support information workers with their daily activities. This prompted the research community to develop progressively open architectures for delivering hypermedia functionality. A number of hypermedia systems have been developed over recent years, each attempting to embody the tenets of open hypermedia. In this chapter a number of the influential systems are described, classified in terms of the Flag Taxonomy [150] and examined against the criteria proposed by Davis [48]. A review of Microcosm is omitted as it is covered in depth in chapter 4.

3.2 World Wide Web (WWW)



Embedded Link

The World Wide Web (WWW) [16] was originally conceived with the aim of providing simple transport and presentation of electronic documents for the scientific community at CERN in Switzerland. The WWW has enjoyed phenomenal growth over recent years, accounting for a significant proportion of all Internet traffic and with a server at almost every site. The WWW adopts the familiar architecture of multiple servers and heterogeneous clients, but the distributed aspects of the system will be examined more thoroughly in chapter 5.

The client applications, such as Netscape Navigator and Mosaic, can be considered open in terms of the standard IETF protocols they support, such as FTP [21] and Gopher [3]. Recent developments from Netscape have seen an enhanced client that provides support for electronic mail and Usenet news through the POP [141] and NNTP [122] protocols respectively.

Any media type can be downloaded via the WWW, but if the browser does not have integral support for the media then a third party *helper* application can be launched. Such helper applications are usually not able to integrate with the WWW to provide any hypermedia functionality, thus leading to a navigational "dead-end".

Additional flexibility at the client can be gained through the use of proprietary mechanisms from Netscape called *plug-ins*. An API[1] is supplied which provides a greater degree of control over the rendering of information upon the client application window. This can be exploited to achieve a customised presentation of the material, as has been demonstrated by Macromedia ShockWave and many others.

As well as being a proprietary solution, the disadvantages of *plug-ins* are that they are platform specific and must first be downloaded and installed prior to use. The Java language from Sun [171] is an attempt to address many of the issues associated with providing dynamic content and greater flexibility in a platform neutral fashion. Java is an object-oriented programming language that is compiled to byte code and processed by a secure interpreter. Sun has licensed the technology enabling Java interpreters to be embedded within WWW browsers, thus allowing Java *applets* to be downloaded and run. Various academics and system administrators have expressed concern regarding the safety and wisdom of executing unknown programs, but the advent of Java has nevertheless inspired much frenetic activity within the Internet community.

In order to accommodate these additional features, the WWW clients, in particular Netscape, have become increasingly large and monolithic. Perhaps such applications would be better served with a component-based architecture. Sun demonstrated how improved versatility and extensibility could be provided in the form of the Hot Java browser [170]. Entirely written in the Java language, the browser can be extended

[1] Application Programming Interface

to manage new protocols and new media formats in a dynamic way.

Potential for flexibility in the WWW also exists at the server, which can be achieved through the Common Gateway Interface (CGI). It is possible for a client to send information to a WWW server whereupon a script is executed that returns an appropriate response to the client. These scripts are typically written using a shell command language or scripting languages, such as Perl.

Hypermedia links are implemented through embedded markup within WWW documents. This format for WWW documents is called Hypertext Markup Language (HTML) [17], and has recently become a standard (see 2.6.3). HTML is based upon SGML[2], and it provides the syntax for influencing document presentation and delineating link anchors together with their binding to a destination document location.

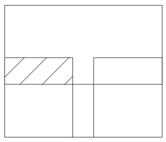
Despite its unparalleled success and the flexibility described above, there are a number of problems with the way in which the hypermedia functionality is achieved. The intrinsic reliance upon HTML violates the principle that documents should remain in their native format. As no separation between structure and content exists, this means that modifying a link requires a change to the document. This is clearly a disadvantage if a link requires alteration to a document that is either stored on read-only media, or for which the user does not have write permission.

The WWW precludes hypermedia navigation from taking place within documents stored in formats other than HTML. In its current form the model also prohibits alternative and multiple webs of links from being applied to the same document in another context, thus limiting the potential for reuse of documents. As the link information is distributed throughout the documents, such a model hampers the provision of tools for visualising the hypermedia network structure. The use of embedded links also makes the task of incorporating third party linking, information retrieval and search engines difficult.

As the rate of information published on the Web increases, perhaps the most concerning issues will be those of management. Whenever a document is deleted, modified or relocated the system is not able to tell which other documents refer to the document in question. This is currently an active area of research.

[2]Standard Generalised Markup Language

3.3 Intermedia



Link Server

Academics at Brown University have been experimenting with electronic document systems since the late 1960's [190]. In 1985 the Institute for Research in Information and Scholarship (IRIS) was established within Brown University and over the next five years this group developed a pioneering hypermedia system called Intermedia [96]. The primary aim of the Intermedia project was to provide an environment for building and delivering electronic teaching resources, but as additional tools were developed it was recognised that this technology had the potential for wider application.

The authors believed that users would be most likely to exploit hypermedia services if they are able to create and follow links as part of their regular working practices. To model how such ubiquitous functionality would look when integrated at the system level, many of the familiar desktop tools were purpose-built and augmented with hypermedia functionality. The authors state that this strategy afforded the impression of *seamless integration* of the application editors with the desktop hypermedia environment [189].

Intermedia was intended to model a multi-user hypermedia framework rather than a monolithic hypermedia program. The design approach with Intermedia was to build an object-oriented framework [139] within which all the dedicated presentation applications could be rapidly implemented and incorporated [118, 121]. An overview of the architecture and framework can be appreciated in figure 3.1. This architecture appears to the operating system as two separate processes. The Intermedia process is the end-user application and the other the link server process. These processes communicate via UNIX sockets, alluded to in section 6.5.

The team demonstrated that by encapsulating the general system-wide policies within the Intermedia framework maximum consistency across all components can be achieved. In an attempt to fit the link-making process into a conceptual model already familiar to users, the act of making links between Intermedia documents was

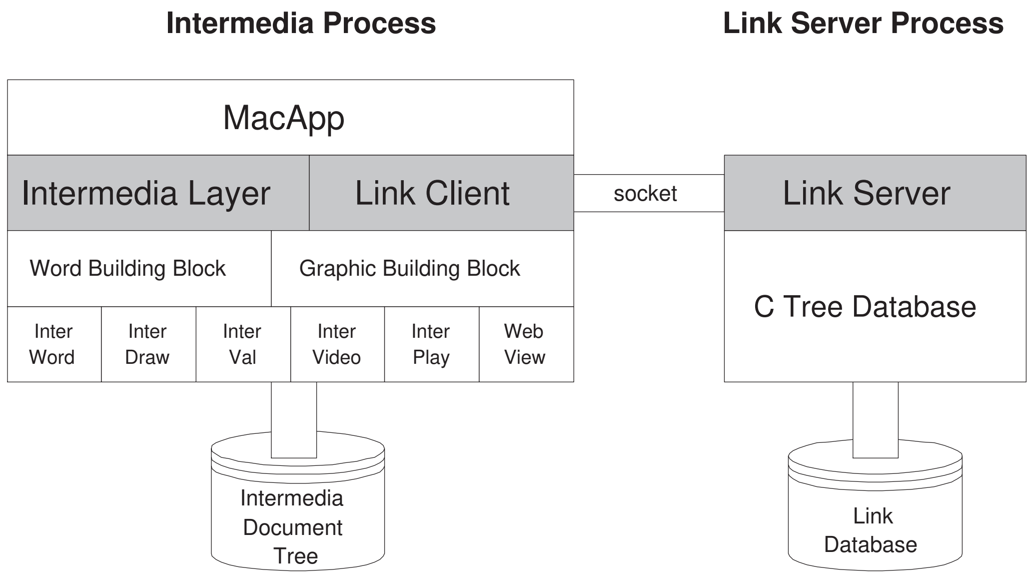


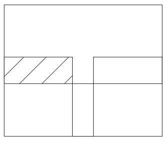
Figure 3.1: Intermedia architecture.

modelled as closely as possible to the Macintosh cut and paste paradigm. Operations such as cut, paste, copy, undo and redo could be applied uniformly to selections, links and anchors.

The designers placed great importance upon keeping document data consistent with the anchor/link data at all times. If file permissions allowed, the users could modify the document content through the Intermedia editors. Although the link and anchor information was held within a separate link database, to preserve integrity in the face of editing operations, Intermedia also stored anchor information within the documents. The concept of a set of links, or an Intermedia *web*, was created to provide a specific context in which related anchors and links could be grouped. Upon reflection, the authors realised that only being able to apply one web at a time to a collection of documents was a limitation. No distinction was drawn between the author and reader, allowing links to be created and followed by all users. Malcolm *et al* [135] called for the provision of templates as an aid to authors when replicating structures of linked documents. The Intermedia team were also conscious of this requirement and developed a solution [40].

Intermedia ran on a dialect of UNIX for the Apple Macintosh (A/UX 1.1), which meant that the system could exploit various operating system features such as file permission controls and the distribution of information. Intermedia was academically successful and very influential, but the selection of a non-standard operating system over the native Apple system led to the premature demise of the project when support for A/UX was discontinued by the vendor.

3.4 Sun Link Service



Link Server

The first implementation of an open hypermedia link service was described by Pearl [154] and heralded as a groundbreaking initiative. The author advocates a loose coupling between the management of data and the management of links, with the link and node information stored separately. This division of responsibilities allows the editing and manipulation of the data to be managed by independent applications, leaving the linking functionality to be provided by a separate, but shared, back-end process. It is important to note that this is achieved without relying upon document markup.

The link service includes a protocol specification, a link server program, a library for integrating new applications and utilities for managing the link databases. It was specifically developed to operate in a distributed environment of Sun workstations. The architecture can be appreciated in figure 3.2. In defining a simple and unrestrictive protocol the autonomy of participating tools was able to be preserved, while providing an acceptable level of integration. Applications can integrate with the link service with minor modifications and the use of a library. The library implements the simple protocol between the link service and the application.

By using such an approach existing tools can be integrated and imbued with hypermedia functionality. This has the advantage that users can access the new services from within their familiar applications. By using existing applications as a vehicle, access to hypermedia functionality can be made available within the user's environment

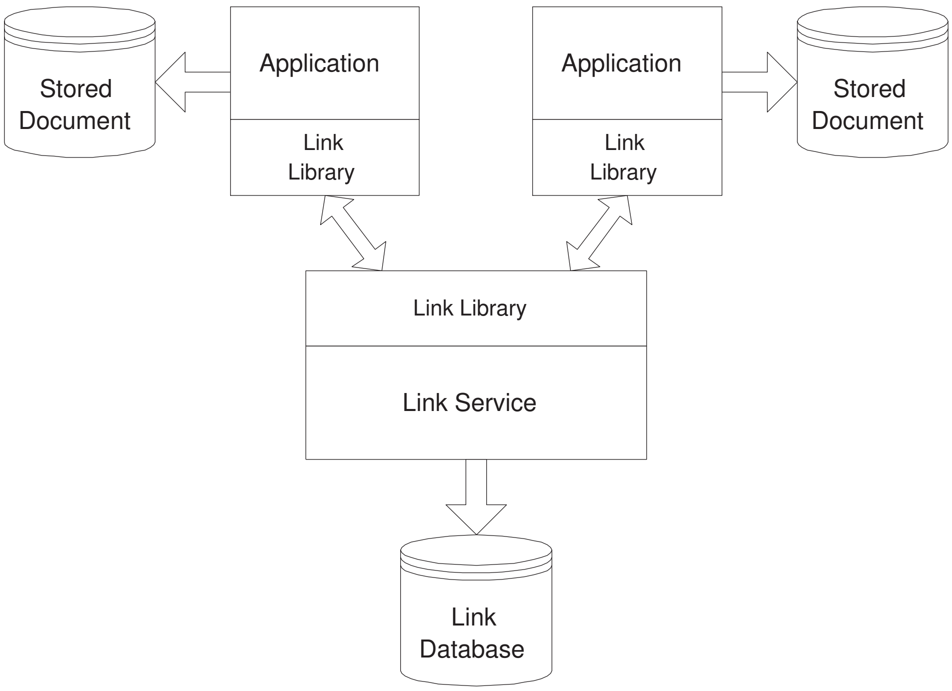


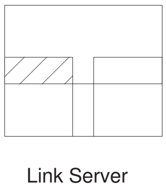
Figure 3.2: Sun Link Service architecture.

in a relatively transparent way. As the number of link service enabled applications increases, it is possible to envisage an integrated information environment. However, when integrating an application with the link service, the programmer has ultimate control over the way in which selections and services are managed within the application. This can lead to the development of inconsistent user interfaces.

The basic link service provides only untyped links with no attributes or directionality. New media can be supported without the need to modify the link service as it is the remit of participating tools to display and generate anchor representations to be stored in the link database. There are no facilities for extending the functionality provided by the link service. The author recognised the issues of maintaining integrity and version tracking as problematic within an open architecture.

Although the hypermedia functionality provided by the link service is relatively simple, the architecture of the system set a significant precedent and has had a profound effect on subsequent open hypermedia systems.

3.5 Multicard



Multicard [159] is an open hypermedia system developed at INRIA in France as part of the ESPRIT Multiworks project. The various components of Multicard include a developer's toolkit, authoring tools, a scripting language, a communications protocol and various editors. The aim of Multicard is to bridge the void between the features offered by hypermedia systems and those required by large industrial applications. It has been demonstrated that by offering a powerful back end system and an open protocol, serious applications can be integrated to provide a unified hypermedia environment.

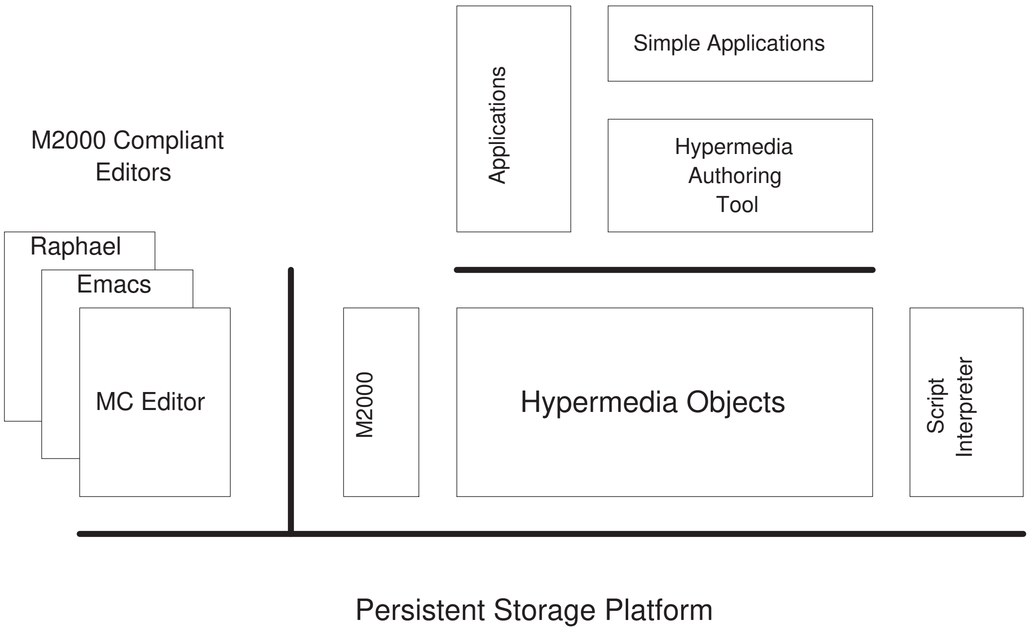


Figure 3.3: Multicard architecture.

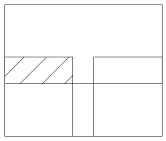
Multicard has a modular architecture and is loosely based upon the Dexter reference model. A toolkit is supplied with a range of classes for managing hypermedia objects such as groups, nodes, links and anchors. An API provides facilities to specific hypermedia editors, tools and general applications for the creation, manipulation and deletion of hypermedia objects. An abstraction allows the designers to use either an object-oriented database or a relational database to provide persistent storage for the hypermedia objects.

A communications protocol, called M2000, defines the interaction between Multicard and the suite of compliant editors which respect this protocol. M2000 is more developed but not as simple and unrestrictive as the protocol used for the Sun Link service [154]. Editors, however, may choose to comply with M2000 at any depth, depending upon the level of integration required. Multicard can demand that an application open and close particular nodes, while an application may inform Multicard of certain mouse events and menu actions. An overview of the architecture can be seen in figure 3.3.

A separate utility, the hypermedia authoring tool, must be used for the creation of a hypermedia structure, e.g., nodes and groups (composites). Once in place, the M2000 compliant editors are used to navigate the structure and edit the content. This two phase process highlights a clear distinction between author and user interfaces.

Multicard views links as event/message communication channels between two end points. It is argued that by allowing links to communicate with scripting languages the re-configurability of hypertext applications can be drastically improved. A comprehensive scripting language attempts to bring the API closer to the user. Node, groups and anchors can all be annotated with scripts and subsequently extend their behaviour.

3.6 PROXHY



Link Server

PROXHY [119], an acronym for the prototype implementation of an extensible hypertext system, is characterised by an object based process model. The authors were motivated to pursue an open and modular design philosophy [154, 140], in which applications may participate in the architecture at various levels.

The simple data model supported by PROXHY is such that anchors connect application objects (node content) to links. Links and anchors together are used to represent associations between application objects in a hypertext. PROXHY distinguishes the structure from the content and uses a persistent object store to record this information.

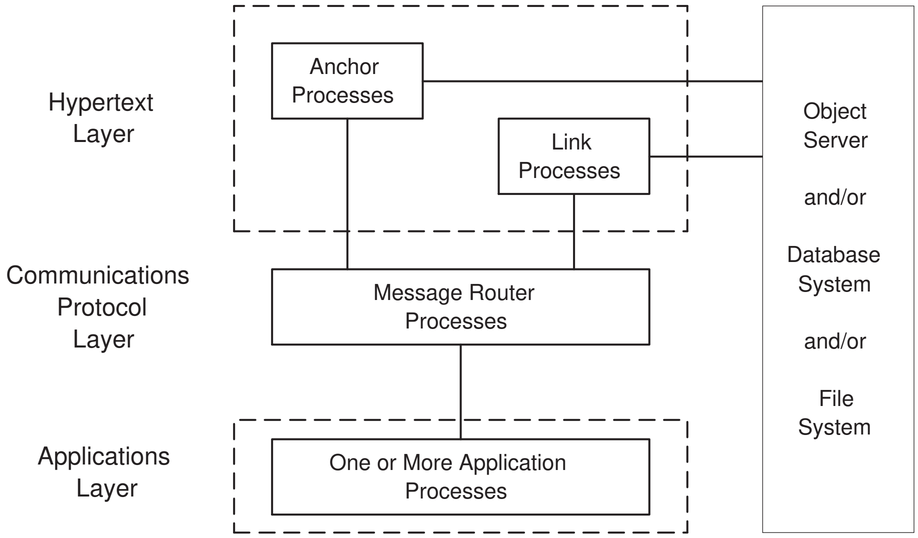


Figure 3.4: PROXHY architecture.

The architecture is divided into four distinct layers, which can be observed in figure 3.4. Each layer can comprise an arbitrary number of processes. The authors advocate an object-oriented approach and provide high-level support for both inheritance and message passing. The communications protocol layer is based upon one or more message routers which are responsible for passing messages between components. The message routers employ object definition tables to coordinate the transfer of messages to the appropriate destination component. Inheritance is also supported by the message router through the superclass and subclass entries in the object definition tables. If the intended target object does not support the method specified in the message, the tables are traversed to locate the parent class.

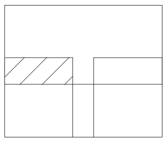
The anchor and link classes which comprise the hypertext layer are implemented as a set of communicating processes. The process model enables this collection of autonomous processes to operate concurrently upon the hypertext structure. By subclassing the anchor and link classes it is possible to extend the functionality or modify the behaviour of these elements.

The back end layer provides a general abstraction to storage services. These services could be provided underneath by an object server, a database system, or a set of modules which utilise the file system.

Building on top of the previous three layers is the application layer. To fully integrate with the rest of the system, an application must be able to understand and process a minimum number of messages. This set of messages pertain to the management and presentation of anchors within the application. As mentioned earlier, applications can participate in the architecture at various levels and by supporting a subset of the messages partial integration can be achieved. However, a limited degree of integration is also possible for applications that cannot be modified through the use of *proxy* anchor objects. During the link creation phase the proxy anchor prompts the user to provide specific details regarding the application. This enables the system to recreate the state of the application following subsequent traversals of the link.

It is clear that the architecture of PROXHY is very flexible, but the volume of network traffic generated by the exchange of messages has been shown to have a serious impact on performance.

3.7 Chimera



Link Server

Chimera [6] is an open hypermedia system developed by a team at the University of California, Irvine. The primary aim of this project is to investigate how hypertext can benefit the specific process of software development. The authors note that software development environments (SDE) are used to develop and maintain a diverse collection of highly interrelated objects. Chimera has been developed to facilitate the capture and visualisation of the relationships within an SDE.

The Chimera data model is similar to that of most Dexter-based systems. One particular distinguishing feature is that multiple views upon a single object can be created, allowing different aspects of the object to be focused on. This additional level of abstraction is achieved through the introduction of *views*. The authors contend that the Chimera data model cannot be completely modelled in Dexter. They argue that Chimera views cannot be represented by Dexter composites, or indeed even specified. Chimera also supports the creation of dangling links.

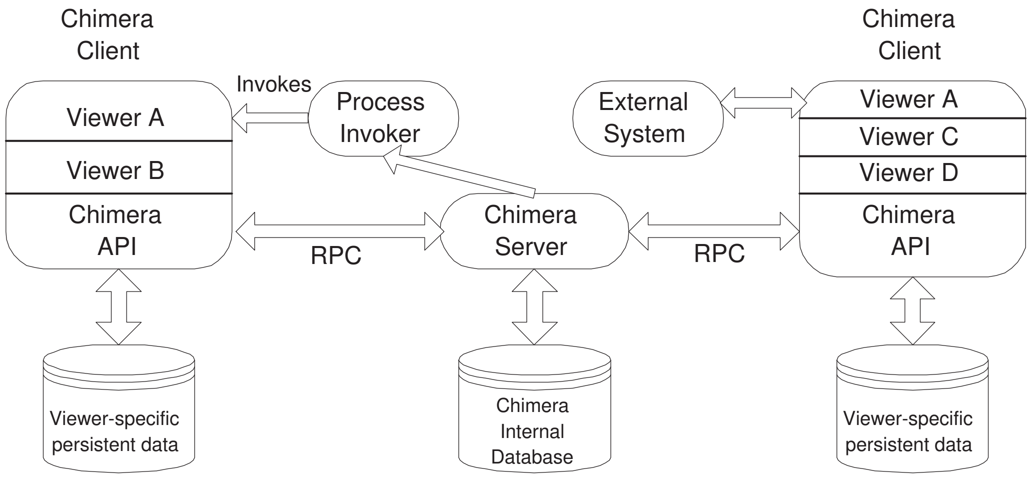


Figure 3.5: Chimera architecture.

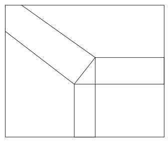
The Chimera architecture is based on a client-server model with the prototype running on Sun workstations. The system architecture can be appreciated in figure 3.5. Multiple clients communicate through an API with the centralised Chimera server. The Chimera server supports a fixed set of hypertext events which any Chimera client can declare an interest in. The server is responsible for managing each connected client by maintaining a record of the viewers currently running, the particular view each viewer is showing and the events each viewer wishes to receive. The Pleiades object-oriented database assists the Chimera server in managing the persistence of link databases, or Chimera *hyperwebs*.

Each Chimera client includes one or more viewers and the libraries necessary to communicate with the Chimera server. The interpretation of *object*, *view* and *anchor* semantics are devolved to each specific viewer. Since each viewer may choose to implement this functionality in different ways, a uniform interaction style cannot be guaranteed. Each client is also responsible for the persistent storage of viewer-specific data. No distinction is drawn between an author and a user.

A commercial desktop publishing application (FrameMaker) has been integrated with Chimera. A wrapper process translates between Chimera hypertext concepts and FrameMaker's internal hypertext concepts. Custom FrameMaker macros were written to enable link traversal messages to be sent to the wrapper process.

Chimera meets the majority of the requirements for an open hypermedia system, but the architecture of the Chimera link server is not inherently extensible as it supports a fixed set of events.

3.8 Hyper-G



Open Hyperbase

Hyper-G [137, 123] is a continuing project at the Technical University of Graz in Austria. It is a large-scale project with approximately sixty members involved in related research and development. Hyper-G is a large-scale, distributed, open hypermedia system. The system is decomposed into a UNIX based Hyper-G server and a client, available for both UNIX and Microsoft Windows. The distributed aspects of the system will be analysed in more depth in chapter 5.6.2.

Hyper-G encourages users to arrange their documents in a hierarchical manner. Documents can then be grouped into aggregate collections, which may in turn belong to other collections. Keywords can be assigned to documents which may be used in searches, or to aid in the automatic generation of links. An integrated search engine allows attribute and content searches to be conducted across the body of stored information.

Originally, before a document could be imported into Hyper-G it had to be converted into a proprietary format called Hyper-G Text Format (HTF), but this is no longer the case. Users are now expected to use public domain utilities to convert their data into HTML. The translated version of the user's document is then imported into the Hyper-G object-oriented persistent storage manager.

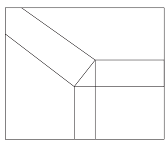
As Hyper-G is a multi-user system, each registered user has their own private information space, analogous to a UNIX home directory, which is simply another collection. All objects in the Hyper-G system conform to a permission system based upon the UNIX model. This allows collections, documents and links to be secured in a consistent manner. The separation of link from node data is also upheld within Hyper-G. The range of link types are limited, with only point-to-point links supported. There is no support for creating separate webs, or link databases, restricting an author from reusing resources in a new context.

A location in the centre of Europe and having an international team possibly provided the initial motivation for incorporating multiple language support within Hyper-G. The Hyper-G design incorporates a plethora of translators to make links automatically available regardless of the original language that the particular keyword was selected in.

The native Hyper-G client is composed of a session manager and a selection of purpose built media editors/viewers. Editors are available for text, browsing the document collection, various graphics formats and a navigation aid. Hyper-G does not distinguish between the author and the user.

The design of Hyper-G meets many of the requirements for an open hypermedia system, although the server architecture is not inherently extensible and therefore may not easily be tailored to individual user requirements. A high level of integration has also been achieved with external information systems such as WWW and Gopher.

3.9 DeVise Hypermedia (DHM)



Open Hyperbase

Since 1991 a group at Aarhus University in Denmark have been working on an object-oriented application framework, called the DeVise Hypermedia framework (DHM) [90, 91]. The aim of this project is the production of general tools to support system development and cooperative design in various application areas, with particular emphasis on information management for large engineering projects.

The designers cite the formulation and specification of the Dexter Hypertext Reference Model as a landmark event in the evolution of hypermedia research. A hypermedia development framework and prototype applications have been developed within the DeVise project for implementing and experimenting with extensions to the Dexter Model. In the course of their work, the authors noted a number of areas in which the Dexter model proved deficient and required clarification and extension.

The Dexter model prohibits dangling links, but the authors argue for the relaxation of this constraint and present four cases where their use can be justified [90]. The authors also offer a more precise definition for anchors and the semantics of link directionality. The DHM data model supports bi-directional links with multiple endpoints, but has recently been updated with the goal of capturing other successful post-Dexter hypermedia systems. Through the introduction of location specifiers (locspecs) and reference specifiers (refspecs) the authors have shown how the Dexter model can be extended to support WWW embedded links and Microcosm generic links in an intuitive, integrated and object-oriented manner [92].

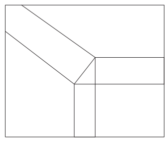
Grønbæk *et al* also acknowledged the need for composites but recognised that the concept could be extended by supplying a set of generic composite classes which can be specialised to support a rich variety of structuring mechanisms [86]. Tabletop composites, atomic composites and link composites are all samples constructed from these base classes.

The DHM system currently supports four atomic component types which are text, file, graphic and movie components. Internal text and draw components provide simple text and drawing editors which support linking spans of text and individual graphical objects. A movie component supports linking to digitized video and current work is addressing linking to specific video segments. Significant success with integrating office and CAD applications has also been demonstrated with DHM.

Support for cooperative work with DHM has also been investigated [88, 89]. A multi-user object-oriented database is used as the basis for providing cooperation support for shared hypertexts. The database supports long-term transactions, flexible locking and event notification pertinent to the shared material.

The DHM framework is platform independent and application variants have been developed for Apple Macintosh, the most advanced, and UNIX/X-Windows with a PC/Windows implementation under construction.

3.10 System Prototype 3 (SP3)



Open Hyperbase

Since 1991, staff at the Hypermedia Research Laboratory (HRL), Texas A & M University, have designed and built a number of hypermedia systems. The evolving design of their hypermedia system (SP 0–3) [119, 126] continues in parallel with that of their hyperbase system (HB 0–3) [126]. The hyperbase supports the data storage requirements of the hypermedia system.

The SP3 has a process-based architecture provided in several layers of clients and servers, as shown in figure 3.6. The participating applications and the link services manager form the hypermedia layer, while the association set manager, the versioned object manager and the storage manager implement the HB3 hyperbase. Sophisticated support for multi-user collaboration is also present within HB3.

The link services manager provides run-time support for inter-application linking by coordinating the components delivering the hypermedia services. Applications wishing to participate in link services must be able to interact with the link services manager. Participation demands that applications can handle persistent selections and respond to messages that implement hypermedia functionality through a set of callback routines. SP3 also requires applications to use its object manager to store application data; this allows SP3 to support versioning of application data and hypertext data.

A distinctive feature of SP3 is that anchors and links are modelled as processes,

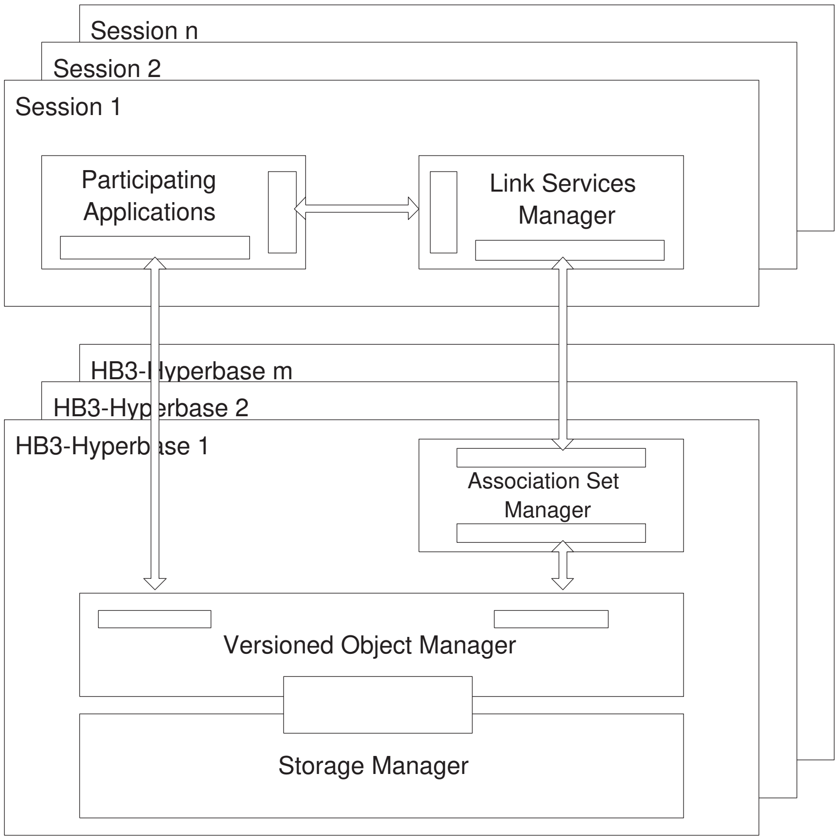


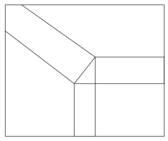
Figure 3.6: System Prototype 3 architecture.

---

first-class executing programs capable of providing a range of run-time semantics. In contrast to the Intermedia approach of embedding policy within the framework, the designers of SP3 devolve the hypermedia policies to the anchor and link processes. The authors claim that a hypermedia reference model, such as Dexter, should be as much process-based as data-based. Like Intermedia, SP3 supports the notion of contexts. But unlike Intermedia, multiple *association sets*, in SP3 terminology, can be applied simultaneously.

By constraining applications to use the HB3 object version manager to store data, it is unclear how easy it would be to integrate third party applications with the system. SP3 is undoubtedly a comprehensive system, demonstrating many of the features required of an open hypermedia system.

3.11 HyperDisco



Open Hyperbase

The HyperDisco [183] project is the product of a joint research effort between members of Aalborg University in Denmark and the Hypermedia Research Laboratory at Texas A & M University. The overall goal of the HyperDisco project is to provide a hypermedia platform for the flexible integration and extension of tools.

HyperDisco was implemented by using the Hyperform hypermedia system development environment [181]. The system currently operates in UNIX environments and is presently hosted on Sun Sparcstations. The HyperDisco architecture is composed of distributed Hyperbase Management Systems (HBMS), tool integrators and participating third party tools as shown in figure 3.7.

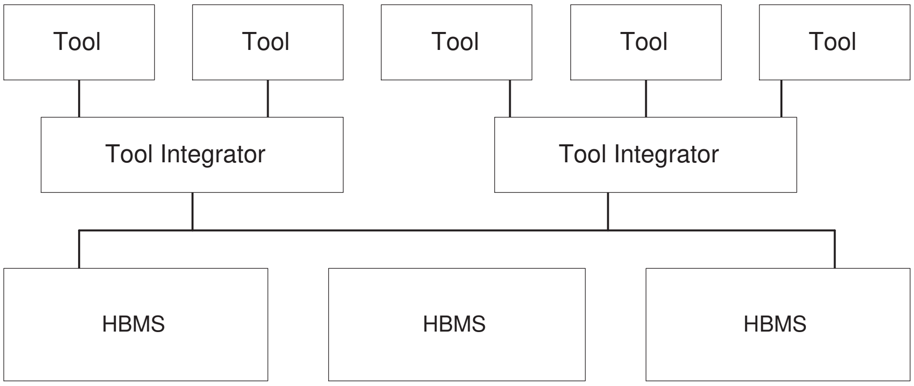


Figure 3.7: HyperDisco architecture.

The tool integrator is the central component of the architecture; it supports multiple integration strategies, allowing various tools to participate in hypermedia services. The tool integrator also controls access and operation of multiple HBMSs and implements all the necessary runtime support for participating tools. Instead of providing a single model of integration that all tools must adhere to, the authors allow each tool, or tool group, to have its own specialised model of integration. An integrated tool may then use as many, or as few, of the hypermedia services as required.

The development of the HyperDisco data model has been inspired by the Dexter model abstractions. The authors have tuned the concepts of components, atoms, composites, links and anchors toward an open hypermedia systems philosophy. HyperDisco provides an extensible object-oriented data model with a set of general hypermedia objects types. These may be specialised through multiple inheritance if so desired. In this way participating tools can extend and tailor the data model by adding new methods and specialising existing methods. Like Multicard, HyperDisco allows scripts to be attached to components and anchors which extend the behaviour of instances.

HyperDisco provides a set of general-purpose built-in services in the form of a library of reusable object-oriented software routines. These classes provide the hypermedia linking services and a flexible communication protocol for the integration model. For the data model the classes provide basic storage for the hypermedia objects.

Sophisticated support for multiple collaborating users is also present within HyperDisco. User controlled locking, concurrency control, notification control mechanisms [182] together with operation tracking allows both the client and the server to recover from crashes with minimal disruption.

3.12 Summary

The majority of the hypermedia systems reviewed in this chapter have been designed to address the issues raised in the previous chapter. Although none of the systems described above meet the all the criteria specified by Davis [48], it is clear that significant advances have been made.

1. *Size*:

   While no explicit upper bound is defined by any of the systems described above, it remains to be seen whether these systems, arguably with the exception of the WWW and Hyper-G, are able to scale well along this dimension. It is becoming clear though that maintaining a WWW site with a sizeable number of documents and links is a non-trivial exercise. By holding the links separately from the documents and replacing absolute URLs with a relative naming scheme the task of managing a large volume of information such as this can be eased. This is the approach taken by the designers of Hyper-G [7] in serving information to the WWW.

2. *Data formats*:

   Intermedia and the WWW both embed anchors within the documents, which renders the documents unintelligible by their original editor and hence reduces them to proprietary formats. Intermedia provided an array of presentation components which supported a rich array of media including video. Although not inherently extensible, the WWW can launch helper applications to view unsupported media. The WWW architecture has no support for temporal media but experimental work on a project called Vosaic [41] has demonstrated how the model can be extended to support real-time video. As HTML is text-based it is feasible to generate WWW pages dynamically prior to display. In this respect the model is not as limiting as first thought. The other systems are extensible and hence able to accommodate new media with minimal changes.

3. *Applications*:

   The designers of Intermedia created an environment in which hypermedia services were available within all the multimedia presentation components. Rather than attempting to integrate with existing applications the designers built these tools from scratch. The WWW also has limited scope for integrating with third party applications.

   The designers of systems such as DHM [90], Multicard [159], Chimera [6] and HyperDisco [183] have employed techniques for achieving closer integration with common desktop applications with varying degrees of success. By delegating the presentation aspects to autonomous third party applications these hypermedia systems are unable to guarantee a uniform user interface to the hypermedia services.

4. *Data models*:

The ability to represent diverse data models within a single hypermedia system and provide a complete and consistent method for interchange between them will remain a challenge for some time. Currently Hyperform [181], and hence HyperDisco [183], is the only example of an open hypermedia environment with a data model that can be customised dynamically. The Hyperform solution is not entirely sufficient though as the base classes offered by Hyperform may not be rich enough to represent every aspect of a desired data model.

5. *Platforms*:

   Implementations of the WWW, Hyper-G and DHM are available for more than a single platform, although it is possible that many of the systems described above could be ported without major modifications.

6. *Users*:

   All of the systems described above can support multiple users, although the WWW treats all users as *anonymous*. While not a pre-requisite for an open hypermedia system, sophisticated support for collaborative work is provided in HyperDisco [183], SP3 [126] and DHM [88].

   While reflecting on the weaknesses of Intermedia the authors recognised the importance of allowing multiple hypermedia webs to be active simultaneously. Of the systems reviewed above Chimera [6] and SP3 [126] describe how multiple webs can be active concurrently.

The following chapter continues by describing Microcosm, an open hypermedia system designed to provide a framework for exploring potential solutions to these issues.

4 Microcosm: A Model for Open Hypermedia

Microcosm is an open hypermedia system developed as a collaborative effort by the Multimedia Research Group at the University of Southampton. Work on the first version of Microcosm began in 1988 with successive versions being implemented to investigate new research directions. In this chapter the aims of the design, the system architecture and some of the main components are discussed. The openness of Microcosm is also examined with respect to the criteria introduced in 2.8.1.

4.1 Project Aims

As mentioned in the previous chapters, by the late 1980's a number of limitations with early hypermedia systems had become evident. It was intended that Microcosm should be an open and configurable hypermedia framework for research into these issues. A detailed account of the initial aims and objectives of Microcosm was first published by Fountain *et al* [66]. Among the main issues outlined were:

Authoring effort required Many hypermedia systems require the links to be created manually, which can be a laborious and very time consuming task. New electronic documents are being produced at an alarming rate and new techniques must be developed for integrating these documents with the hyperstructure in a more efficient manner.

Closed hypermedia systems The limitations of closed systems were covered in chapter 2, whereas a more open architecture can aid greater integration with the desktop, functional extensibility and support for new media.

Proprietary document formats Hypermedia systems in which a document must be converted into a proprietary format prior to use make it difficult for the document to be used in another application.

Embedded links It was recognised that hypermedia systems which embed links within the documents are unable to incorporate and link to information held on read-only media, e.g. CD-ROM.

While not all existing systems suffer from the shortcomings listed above, the aim of the Microcosm team has been to design a new system which addresses these issues by providing an open model for hypermedia with dynamic linking. A set of design principles for Microcosm were also used to further guide the design:

No distinction between author and user Equal access to the linking functionality of the system should be granted to all users. The ability to create links should not be viewed as a privilege restricted only to teachers or system administrators.

Loosely coupled system with minimum interdependencies The system should not be a single program, but instead composed of a set of communicating tasks. By reducing the interdependency between individual tasks it should be possible to integrate other applications with the system.

Modular architecture to allow replaceable components As Microcosm is to be used as an aid to hypermedia research, the architecture must be flexible enough to accommodate new components and allow them to be configured dynamically.

Separation of links from the documents By maintaining a separation between content and structure, new tools can be developed to analyse and manipulate link information.

In addition to the principles outlined above, the design of Microcosm was driven by the adoption of a *selection/action* metaphor as the primary means of user interaction. For example, the user selects an area of a document (e.g., words in a text document, or a portion of an image) and then chooses which action to apply to that selection. These actions may vary from following a link to initiating some complex image processing. This approach allows the following of a link to be viewed as just one of a number of alternative methods for navigating through the information space.

4.2 The Conceptual Model of Microcosm

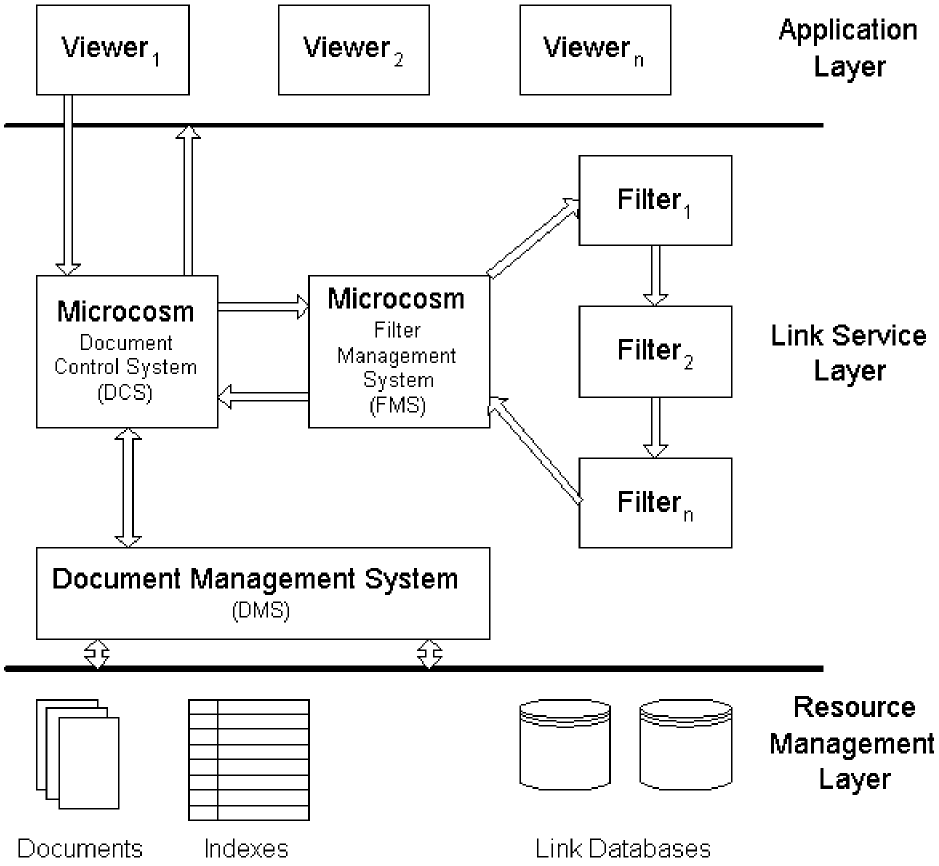


Figure 4.1: The conceptual model of Microcosm.

In a similar fashion to the Dexter Model, the conceptual model of Microcosm can be sub-divided into three layers. The *Application Layer* comprises a number of multimedia presentation programs, or *viewers*, each able to render one or more media types. The *Link Service Layer* provides an extensible range of hypermedia services in the form of *filters*, for example, link creation, link following and full-text retrieval. The *Resource Management Layer* is responsible for maintaining the documents and various indexing data used by the layers above. The model is illustrated in figure 4.1.

4.3 The Microcosm Architecture

The development of Microcosm version 1.0 enabled proof of concept to be established, but the prototype was significantly re-designed for the second version. Although various members of the research group have developed additional components for Microcosm, the core architecture has survived almost unchanged since version 2.0. Consequently, the focus of this discussion will relate to version 2.0 and above.

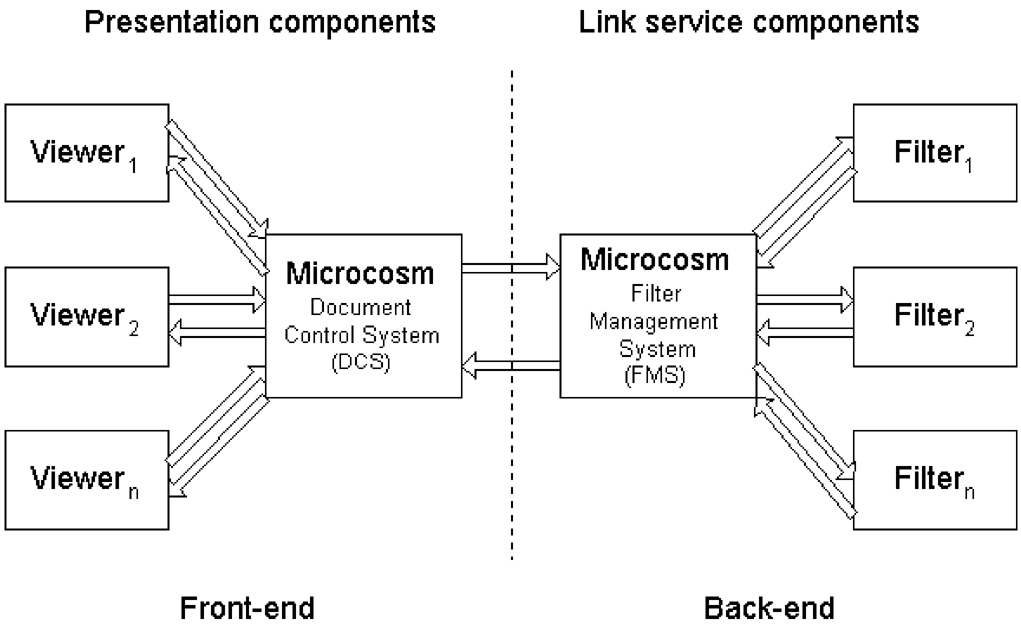


Figure 4.2: The Microcosm architecture.

The architecture of Microcosm was first described by Fountain *et al* [66]. It is best understood as a set of autonomous communicating processes which supplement the operating system facilities. Microcosm has a symmetrical architecture: the front-end of the system is responsible for the presentational aspects and is managed by a process called the Document Control System (DCS), while the back-end provides the hypermedia link services and is managed by a process called the Filter Management System (FMS). Both the DCS and the FMS coordinate the behaviour of a number of child processes which, acting together, deliver the functionality of the system. An appreciation of the Microcosm system architecture can be gained from figure 4.2.

4.3.1 Loosely Coupled Independent Processes

On the design of modular systems Thompson [173] asserts:

> The key point ... is that it is modular and open. This modularity is based on the observations that the functions the modules perform are independent of each other, that is *orthogonality implies modularity*.

Shackelford [165] notes that future hypermedia systems will be based on modular architectures that separate orthogonal concerns into plug-compatible components. As orthogonality implies modularity, modularity in turn, implies choice.

The Microcosm system can be described as modular as it is composed of an arbitrary number of communicating processes. This set of processes are unified by a communication subsystem which provides the necessary primitives for allowing messages to be passed between them. A message format for structuring the communication between the processes is also required.

4.3.2 Inter-Process Communication

In an attempt to avoid reliance upon platform specific transport mechanisms, communication in Microcosm is based upon the concept of an abstract channel. Once a channel has been established between two Microcosm processes communication can take place. Each document viewer establishes a channel to the DCS and, similarly, each filter establishes a channel to the FMS. Messages are passed between the front end and the back end of the system via a channel connecting the DCS and FMS. The connections between the various processes in the architecture can be seen in figure 4.2.

The abstract channel approach is embodied by a shared library which provides an additional layer above the native inter-process communication mechanism. The library provides a simple and convenient approach to message passing. Before a new process can exchange messages with the rest of the system it must inform the library of its main window handle. This helps the library to route messages to the appropriate processes. The library then issues the new process with an opaque identifier which is used for all subsequent communications activities. Once subscribed to the communications library, a process can send and receive messages freely.

The communications library in Microcosm version 1.0 used DDE[1] as its messaging protocol, but it became evident that it was not well suited to the new flexible Mi-

[1] Dynamic Data Exchange is the Microsoft Windows proprietary protocol for inter-process communication mechanism

crocosm message format. The underlying system was modified to use the Microsoft Windows basic model for message passing. A detailed description of the development of this library is given by Heath [107].

4.3.3 Extensible Message Format

The unit of communication between Microcosm processes is the Microcosm message. In Microcosm version 1.0 the message structure was of a fixed nature. Each message had fields containing the source of the link, the selection and the action to be performed. The fixed nature of the message format meant that as new modules for Microcosm were developed which required additional fields in the message the system had to be updated. As this was clearly undesirable, a new structure for messages was devised.

A Microcosm message is a flexible and extensible abstract data type with a suite of functions allowing processes to manipulate the contents of a message. Concealed beneath the abstraction, a Microcosm message is simply a text string comprising an arbitrary number of tuples, each containing a tagged name and an associated content body. The message is based on a simple plain text encoding which makes it easy to implement and interpret both by Microcosm and third party applications. A typical Microcosm message can be seen below.

```
\Action CREATE.LINK
\Selection Bridget Bardot
\Offset 36
\DocID 100.96.10.25.10
\Date 12.10.1996
\Description Famous French actress and animal rights activist
```

A shared library was developed which provided a number of constructor and accessor functions for building and manipulating Microcosm messages. Although it is a non-trivial operation to concatenate a number of string tuples together to build a message, or to search the entire string to extract a tuple of interest, the library provides a number of convenience functions for these routine operations. This library is documented in some depth by Hill [109].

4.4 The Document Control System (DCS)

As alluded to above, the Document Control System (DCS) manages the document viewers associated with Microcosm. The primary role of the DCS is to invoke viewers and to initiate communication with them. As multiple viewers may be active at any given time, the DCS has to maintain a record of each viewer and coordinate the routing of messages between them and the rest of the system.

In a typical Microcosm session a user will interact with one or more document viewers and issue actions such as *follow link*. This results in a viewer parcelling up the necessary information for this action into a message and sending it onto the DCS. The DCS encodes an identifier for that particular viewer within the message and forwards it onto the FMS. If, or when, a response message is returned from the back end of the system via the FMS, it is examined by the DCS to locate the identifier of the originating viewer. The message is then forwarded to the appropriate viewer.

Rather than spawning a new viewer process every time the user wishes to browse a new document, the DCS tries to reuse existing viewers to present the document. A facility for *locking* viewers is also offered by the DCS, which allows a viewer to signal to the DCS that it is not available for reuse. Viewers inform the DCS of any change of state in this respect by sending a message in the same manner as described above.

4.5 The Filter Management System (FMS)

The extensible range of hypermedia services offered by Microcosm are supplied by the filter processes. These filter processes are managed by a control process called the Filter Management System (FMS). By decomposing the core functionality of a hypermedia link service into a number of constituent parts it becomes clear how greater choice can be offered. Through the addition of new filters the functionality of the link service can be augmented dynamically, and, similarly, one component can be replaced by another offering an alternative strategy.

The structure of the back-end of the system mirrors that of the front-end. The most obvious difference between them is that the user interacts with the document viewers and instigates actions, whereas direct user interaction with the filters is less common (typically to set options). These processes are referred to as filters, not in the context of UNIX, but because they intercept messages from the FMS and are capable of *filtering* information based on the content of the message.

The two halves of the architecture are united by message passing. Messages are sent from the viewer to the DCS, which passes them onto the FMS and through a single, sequential chain of filters. Each of the filter processes in turn is given the opportunity to inspect and respond to the message. Based on the content of the message, some filters may add new messages to the chain. Eventually the message(s) will emerge from the chain arriving at the Link Dispatcher. This process examines the messages to see if they contain any available actions (such as links to follow), and if so presents these actions to the user. A comprehensive examination and analysis of the FMS can found in chapter 6.

4.6 Document Viewers

A document viewer is responsible for providing an interface for allowing the user to manipulate the medium, or media, and also to perform actions such as create and traverse links. A number of dedicated viewers for Microcosm provide support for media such as text, images, video and audio. The system can be augmented to support additional media with minimal overhead, which is largely attributable to the open design of the system.

In a perfect world the current desktop applications would serve as viewers for the hypermedia link service. As alluded to in chapter 2, it is in practice a non-trivial task to achieve this ideal. It is clear that applications differ in their ability to be integrated with the link service. Microcosm supports viewers with different levels of awareness of the link service protocols. These viewers can be classified into three broad categories, each of which are described in the following sections.

4.6.1 Fully Aware Viewers

A fully aware document viewer is a program that has been specifically developed, or modified, for use with Microcosm and subsequently has the ability to communicate with the system using the established protocol. Messages can be passed freely both to and from the rest of Microcosm, facilitating the full integration of the viewer. Consequently, the viewer can exploit the entire range of services available from Microcosm. Figure 4.3 is a screenshot showing a fully aware text viewer in use. As the techniques for integrating third party applications with Microcosm are improving, the requirement for proprietary fully aware viewers is diminishing.

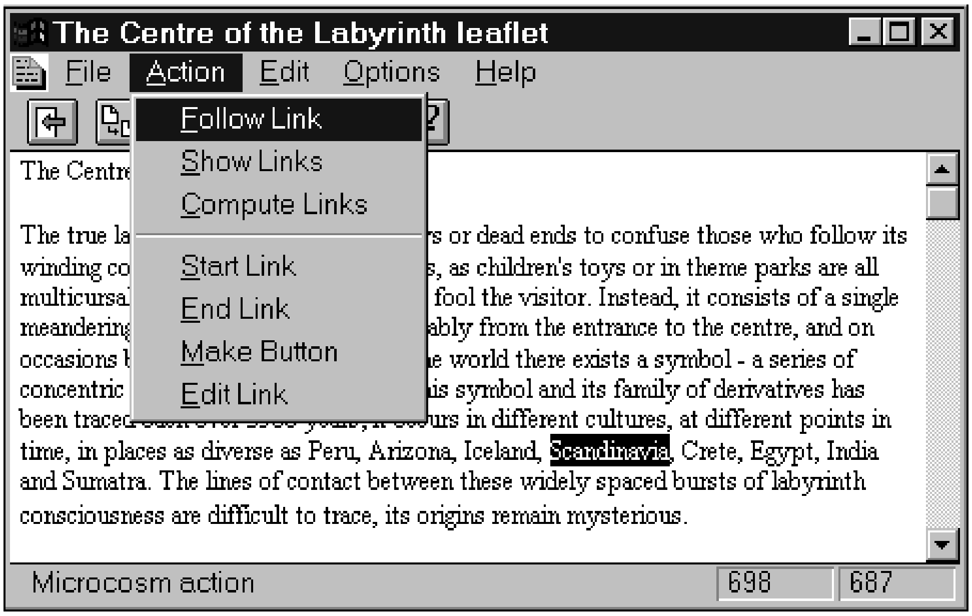


Figure 4.3: Applying an action to a selection within a fully aware Microcosm viewer.

4.6.2 Semi-Aware Viewers

A semi-aware document viewer is an application which is initially Microcosm unaware, but can be modified to become at least partially Microcosm aware. These are applications which typically provide some form of programmability together with methods to allow communication with external applications by whatever means available. It is common for commercial applications to allow some degree of tailorability through the use of a macro language. This makes it possible to add extra menu options and communicate with other applications with minimal effort.

Examples of applications which have been successfully integrated with Microcosm using the techniques described above are Microsoft Word, Microsoft Access, Autodesk AutoCAD and Lotus 123.

4.6.3 Unaware Viewers

Unaware document viewers are normally third party applications, not modified in any way, but used to display resources on behalf of Microcosm. They are invoked as a separate module by Microcosm but then exist outside of the Microcosm framework. With minimal coercion this class of viewer can harness the power of generic links, providing that selection-based information can be transferred from the application to Microcosm. By exploiting features of the GUI[2] to expedite the sharing of data between applications, such as via the clipboard, a Microcosm unaware application can access a subset of the hypermedia functionality.

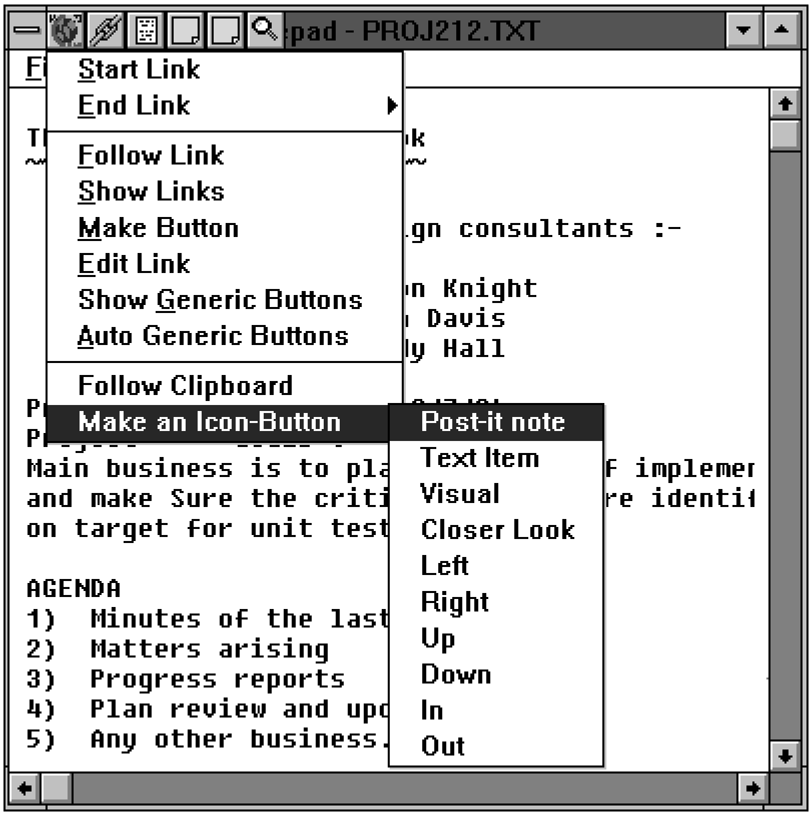


Figure 4.4: Making hypermedia services available from within an unaware application by adding functional icons to the title bar.

Recent work targeting the issue of integration has been conducted by Davis *et al* [52, 124], who describe a Universal Viewer, which acts as a mediator between the

[2]Graphical User Interface

unaware document viewer and Microcosm. The Universal Viewer is a parasitic program that spawns a third party application with a document and then continuously monitors the application. The title bar of the application is adorned with miniature icons each providing either menus or actions associated with the link service. This allows the user to access link service functionality from the application but without requiring any modification to it. This approach can be observed from a screenshot in figure 4.4

The notion of a hypermedia link service as an extension of the operating system is an approach currently being explored by Lewis in his Macrocosm system [129]. By using Apple Macintosh system 7.0 Apple Events, he describes how a system level hypermedia link service can be made available to all compliant third party applications.

4.7 Service Providing Filters

As described earlier, a message emanating from a viewer reaches the DCS and is forwarded to the FMS. The FMS then passes this message to each filter positioned in the sequential chain. Each of the filter processes in turn is given the opportunity to inspect and respond to the message.

The hypermedia functionality of Microcosm is embodied within the filters. The central benefit of this architecture is that the user can customise their personal configuration by selecting apposite filters. Subsequent filters may be started and included in the filter chain at any time.

Presented below is a brief review of the broad range of functionality that can be provided to the user through the open framework of Microcosm.

4.7.1 Linker Filter

The linker filter intercepts pairs of messages with *start link* and *end link* actions. These messages are normally sent from the viewers and contain detailed anchor information. A dialog is then presented by the linker filter in which the user enters the appropriate characteristics of the link they wish to create. Standard information such as the link type and description can also be supplemented with special attributes applicable only to temporal media. The Microcosm model has been extended to accommodate temporal links [83, 84], which are automatically followed when triggered by an active anchor [152].

Once these details have been captured by the dialog, the linker encapsulates the information into a *create link* message and passes it along the filter chain for persistent storage within a link database.

4.7.2 Linkbase Filter

The primary role of a hypermedia system is to facilitate the navigation around a body of multimedia information through the traversal of links. As expounded in previous chapters, the abstraction of the link information from the documents into one or more link databases can yield distinct advantages. A link database, or linkbase, filter is responsible for fulfilling the traditional tasks of creating new link entries within the database, resolving follow link actions and returning all of the links for a specified document.

A linkbase filter is positioned in the filter chain such that it can receive these particular request messages, process them and send on any resulting messages. The linkbase intercepts a message with a *create link* action from the linker, extracts the relevant fields from the message and stores the information. When the linkbase receives a message with a *follow link* action the source selection information is extracted from the message and used as the argument to the resolver function, which, in turn, dispatches a set of references to destination documents. A viewer can also requisition all of the links for a particular document from the linkbase filter by issuing a message with a *find buttons* action. A detailed description of the design and implementation of the linkbase filter is available [107].

Microcosm currently supports three primitive link types, outlined below:

Specific links are defined on an object at a *specific* location in a *specific* source document. It is possible for the location of the selection object to be expressed as a byte offset, a set of vectors or a named region.

Local links are defined on an object at *any* location in a *specific* source document. A link can be followed from the source document regardless of the location of the selection object.

Generic links are defined on an object at *any* location in *any* source document. A generic link can be followed from any document in which the selection object is present.

	Source			Destination		
(a)	Selection	Offset	Document	Selection	Offset	Document
	Bridget Bardot	36	100.96.10.25.10	St. Tropez	78	100.96.08.15.19

	Source			Destination		
(b)	Selection	Offset	Document	Selection	Offset	Document
	Bridget Bardot		100.96.10.25.10	St. Tropez	78	100.96.08.15.19

	Source			Destination		
(c)	Selection	Offset	Document	Selection	Offset	Document
	Bridget Bardot			St. Tropez	78	100.96.08.15.19

Figure 4.5: The internal representation of a (a) specific link (b) local link (c) generic link in Microcosm.

A *button* is simply a special case of a specific link in which the selection object is somehow highlighted within the document source to indicate the presence of an anchor. From figure 4.5 it can be seen that a variety of link types are offered by Microcosm by masking the various fields that comprise the representation of a link. The structure of a link in Microcosm differs from that of a link in a Dexter-based system in that the source and destination anchor information is bound together with the link.

An alternative solution proposed by Beitner [25, 24] is to construct the hypermedia web using a higher level scheme. This requires the author to consider the important *concepts* around which the hypermedia resource base is to be focused, then forge any association(s) between the documents and the relevant concept(s). Advocates of this approach argue that concept and document clustering techniques can produce a more structured, hierarchical hypermedia web expediting less complicated navigation strategies and faster authoring.

4.7.3 ODBC SQL Filter

A significant proportion of corporate information is stored within database management systems (DBMS), usually only accessible through standard query interfaces. As integration with third party applications has proved to be a great strength of Microcosm, the ability to integrate with commercial relational databases was a justifiable pursuit. We wished to allow users to browse and create links on database information in a hypermedia style.

As users interact with Microcosm via a *selection/action* model, any selection can map to any action whose functionality can be provided by a filter in the chain. A useful action may be to perform a database query. This service could be provided by Microcosm in one of two ways. One approach would be to facilitate the binding of a selection to a database SQL query, effectively authoring a link with the result of an SQL query being the destination document. An alternative would be to provide an option to open a dialog box allowing the user to enter an SQL query to be processed, with no requirement for a selection.

Microsoft Access was the commercial DBMS with which this experimental work was conducted [74]. The capability of Access to act as an SQL server via DDE and it being the first ODBC[3] compliant RDBMS available were among the central reasons for its selection.

The prototype is a Microcosm filter that intercepts database query requests from viewers and in response issues a dialog box in which the user enters an SQL query. Once entered, the query is packaged into the specified protocol and delivered to Access via DDE. Upon receiving the request, Access processes the query and returns the resultant table to the filter also via DDE. The necessary justification and tabbing is performed on the resultant table prior to it being saved. The document is then imported transparently into the Document Management System, and subsequently displayed in a text viewer. The user is then able to perform actions upon the table such as following generic links and full-text querying.

With the advent of the *de facto* ODBC standard, it was a logical progression to modify the prototype filter described above to exploit the new technology. The stan-

[3]Open Database Connectivity

dard defines an interface permitting maximum interoperability, allowing a single application to access different database management systems. An application programmer can now develop, compile, and ship an application without targeting a specific DBMS. Converting the filter described above to use the ODBC library proved to be a trivial task, extending the scope of applicability of the filter to any ODBC compliant DBMS.

One engaging aspect to this work was how to manage the transient nature of the result of the database query. Each table produced as a consequence of an SQL query generates a dynamic document and the challenge is to find an appropriate way of managing this document. If the hypermedia system supports document versioning then the decision of how such documents are to be managed can be devolved to those respective components, otherwise the filter must provide an option for configurability.

Using the technology described above it is possible that both the Microcosm Document Management System and the linkbase filter could be implemented in this manner. This approach would allow Microcosm to be viewed as an open link service layer which uses third party desktop applications to provide the presentation layer and any compliant commercial database system to provide the storage layer.

4.7.4 Computed Linker Filter

As an alternative to navigating with links, a filter exists which enables users to issue full-text retrieval searches to locate documents, or sections of documents which share a similar vocabulary. An index of the stemmed keywords of every text document in the resource base is created and maintained by a separate utility. On receipt of a message with a *compute links* action, the computed linker filter extracts the text selection from the message and applies information retrieval techniques to the index to locate related documents. The results are presented to the user in an order according to the statistical significance of the match. The theoretical foundations underpinning the operation of this filter are available [133, 132].

4.7.5 Navigation Filters

The conventional method of obtaining information has been through literature and the book. Books are very familiar to us. They have a title, a contents list, numbered pages, an index, and most importantly they are designed to be read sequentially. We understand clearly how to locate the desired information.

Non-sequential access to hypermedia resources has introduced many problems regarding the navigation and location of desired information. Some approaches to the problem of navigation through a set of documents concentrate upon developing utilities to aid the user in locating their position and then identifying documents of possible interest. A number of navigation devices implemented as Microcosm filters were developed by Wilkins [187]. These include a history list of the documents visited, a local map of the user's current location in hyperspace and a guided tour. But more recently, initial work applying virtual reality within Microcosm [178] indicates that these techniques can offer fresh perspectives upon the problems of hypermedia navigation.

4.7.6 Available Links Filter

The Available Links filter is always positioned at the end of the filter chain and, following a link traversal or search action, displays the descriptions of the destination documents. If this action results in more than one destination document then the list of potential documents are presented to the user to peruse, otherwise the default behaviour is to forward the message to the DCS which invokes a viewer to display the document.

The original Available Links filter has been improved upon significantly by Beitner [24]. Instead of supplying only a textual description of the destination document, an abstract view of the document is generated. In the case of a text document this may be an icon to symbolise the media type together with the description, but this approach is best appreciated for more complex documents. Thumbnail sized graphical icons, moving icons (micons) and audio icons (earcons) can impart more information to the user than textual descriptions alone. Techniques such as this can reduce network traffic significantly if the documents reside on a remote file system.

4.8 Support for Multimedia

As far as support for multimedia is concerned, Microcosm has two proprietary fully aware viewers: an extensible graphics viewer with support for a plethora of graphics formats including BMP, JPG, GIF, and AVI, and an extensible sound viewer [83, 84] providing support for WAV[4], MIDI[5] and CD[6]. A screenshot of the sound viewer can be seen in figure 4.6.

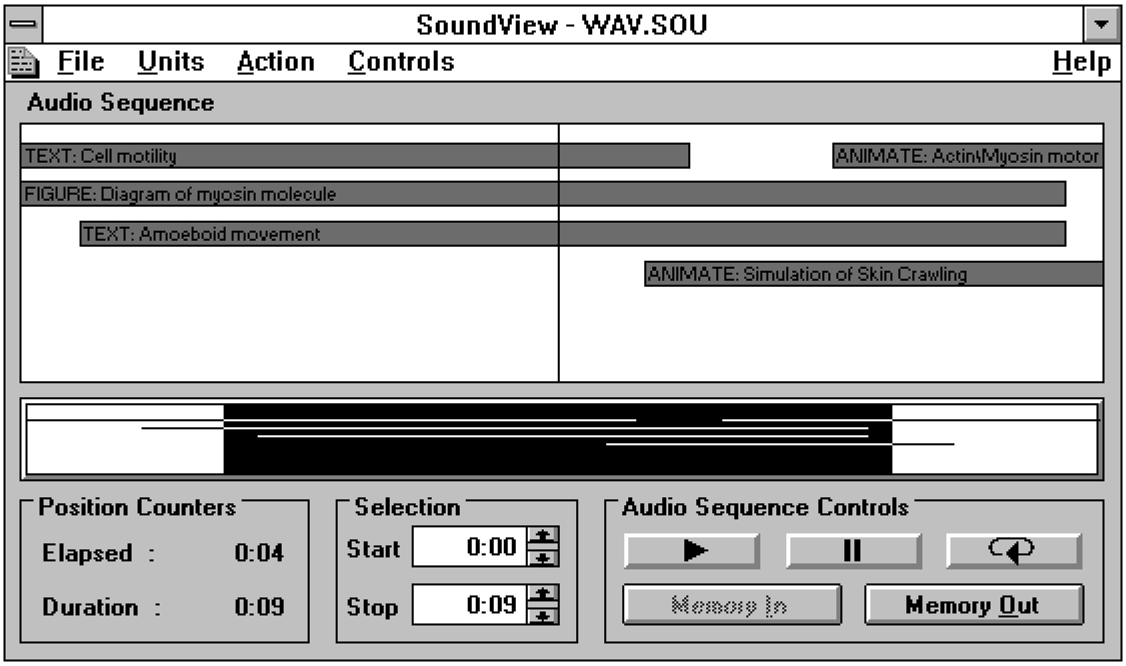


Figure 4.6: The Microcosm viewer for sound.

By developing proprietary fully aware viewers for graphics and sound it was possible to explore a variety of methods for indicating the presence of an anchor within the media. The provision of link anchors for video data is a worthy pursuit and continues to be tackled by various research groups. The approach taken with video in Microcosm [23] allows authors to "click and drag" with the mouse to define a region of interest and then bind it within time and space in the video sequence. As the video sequence plays anchors appear as polygonal outlines moving in time with the video

[4] A proprietary digital audio file format from Microsoft.

[5] Musical Instrument Digital Interface - A standard for controlling electronic musical instruments.

[6] Digital audio Compact Disc

and disappear when no longer valid. This mechanism provides a natural presentation of anchors that remains consistent with the model employed within other Microcosm viewers. While this approach has its merits it only provides link information with respect to the current frame. The user is afforded no overview or visual indication of the presence or relative position of any other anchors within the sequence, such as is provided by the overview window in the sound viewer. It is highly likely that users of the video viewer would appreciate the combined benefits that both representations offer as has been demonstrated with the sound viewer.

Beitner [25] suggests that as the research community has progressed from hypertext to hypermedia insufficient consideration has been given to the underlying structures required to support linking between multimedia. The MAVIS project (Multimedia Architecture for Video Image and Sound) [131, 130] has developed a framework for authoring and following links in non-text media. The generic links work by extracting a variety of representations or signatures from the selection and using pattern matching between signatures at the link following stage.

4.9 The Document Management System

The viewers in the initial version of Microcosm accessed the media directly from the file system, but this approach soon proved inadequate. MS-DOS places a restriction upon the length of filename and provides no means of attaching supplementary information to a file. To overcome these limitations an additional layer of indirection over the native file system was introduced.

The Document Management System (DMS) maintains a database of all the documents known to Microcosm together with any related information a user may wish to store about each file. The single most important aspect of the DMS is that each document is assigned a unique number, as opposed to the original filename being relied upon for identification purposes. Added flexibility is gained by removing any dependence upon explicit filenames within the linking model. A user is able to relocate a document in the file system without endangering the integrity of the linking model by updating a single entry in the DMS. The design issues and implementation strategy are examined closely in 7.9.

4.10 System Configuration

All of the configuration information for Microcosm is stored within a database called the registry. Typical information held within the registry includes the appropriate viewers to use for each media type, the list of filters and associated configurations and the certified users of the system.

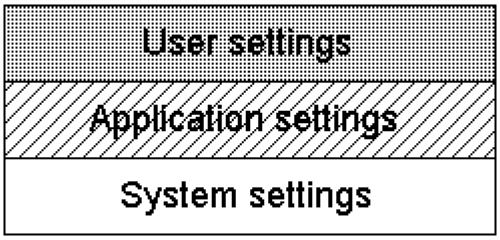


Figure 4.7: The three levels at which Microcosm can be configured.

There are three levels at which Microcosm can be configured, which can be seen in figure 4.7. Specialised technical configuration details are normally stored at the system level. The application level above inherits from the system level, but these settings can be overridden to provide an application wide configuration. Similarly, the user level inherits from the application level, but each user has the ability to override the application specific settings to customise their individual environment. As one might expect, the user settings have the highest precedence.

4.11 Application Areas of Microcosm

The open and flexible nature of the Microcosm architecture has enabled the system to be used effectively within a number of application areas. Briefly described below are the various educational and engineering research projects for which Microcosm has been the enabling technology.

4.11.1 Applied to Education

Microcosm has been used extensively for research and teaching within the History Department at Southampton [42]. Students use Microcosm to investigate various collections of historical material. Once a student is satisfied that they have a good appreciation of the information they can be tested by the system. The HiDES filter is responsible for administering such tests. Once the student has entered an answer, the HiDES expert system analyses the answer and generates a response. The response is a critical assessment of the student's answer and, if not satisfactory, may contain a list of appropriate documents to be re-read. As the generated responses are in the form of documents, hypermedia links are provided to any documents that the HiDES filter suggests should be re-read.

Another teaching initiative in which Microcosm has been used is the SToMP program (Software Teaching of Modular Physics) [12, 13]. A number of physics undergraduate course modules have been developed from a rich set of resources including many dedicated applications to demonstrate key principles.

4.11.2 Applied to Engineering

The aims of the Pirelli project [44, 45] were to assist with the maintenance of complex engineering equipment. The maintenance manuals pertaining to a specific machine were converted into various electronic formats and imported into Microcosm. As, and when, the machine develops a fault the maintenance engineer can use Microcosm in conjunction with an expert system to aid with fault diagnosis, and then for rapidly accessing the relevant technical documentation regarding the faulty system component.

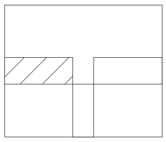
The FIRM project (Factory Information Resource Management), a continuation of the Pirelli project, will seek to extend the assistance beyond that of the maintenance engineer to the entire factory. It is envisaged that Microcosm will be extended to support activities such as stock control, purchasing and quality assurance procedures.

Project management support within the construction industry was the main objective of the Trant project [60]. By integrating a project management tool with Microcosm, it was demonstrated how hypermedia was able to unite the related activities throughout the project lifecycle.

The engineering research projects described above lead to a greater understanding of the requirements of industry and should contribute to the development of an environment such as that described by Malcolm *et al* [135].

4.12 Summary

In this chapter the aims, design issues and versions of Microcosm have been described. From this we are able to see that Microcosm has been successful in meeting all of the initial objectives outlined by Fountain [66]. How open though is Microcosm when compared against the criteria offered by Davis [48] ?



Link Server

No explicit limit is placed upon the size of any object within Microcosm. It has been described how the presentation layer of Microcosm is composed of an extensible number of viewers which support one or more media types. This approach allows Microcosm to support future data formats. Although a number of proprietary fully aware viewers are provided with Microcosm, considerable success in integrating a number of diverse third party applications with the link service has been demonstrated. Like the majority of the hypermedia systems described in chapter 3, Microcosm supports a single data model. It has been shown that the data model is flexible enough to be extended to support temporal links with minimal changes. Three implementations of Microcosm have been developed to date. The most advanced version is for the PC under Windows 95, with other versions available for the Macintosh and various flavours of UNIX. Although the commercial version of Microcosm cannot operate in a distributed environment, a research version of the system that does will be discussed in the next chapter. Multiple concurrent users are not supported by Microcosm, however the preferences and configuration for each user are recorded and preserved for future use.

Modern software system design is an exercise in synthesis, but, as Halasz observed in his 1991 keynote, the research community have yet to agree upon the appropriate decomposition of hypermedia functionality and the protocols for coordinating such modules. The architecture of Microcosm is one such approach for providing an open and extensible link service to aid information management.

The advent of the World Wide Web has opened the eyes of the user community to the benefits of a globally networked information system. The next chapter reviews a number of successful contemporary distributed information systems.

5 Distributed Information Systems

5.1 Introduction

It has been argued in previous chapters that an open approach to the design of hypermedia systems can yield many advantages. One criterion noted by Davis [48] that is of direct relevance to the material in this chapter is that an open hypermedia system should be capable of operating in a heterogeneous and distributed environment. Users are becoming increasingly aware that access to networks, in particular the Internet, greatly enhances the amount of information at their disposal. As the volume of information residing on computer file systems around the world escalates at an alarming rate, the task of designing information systems able to provide adequate management services becomes more difficult. Although distributed hypermedia systems fall within the wider definition of distributed information systems, they offer a fresh perspective on information access, presentation and management. The evolution of distributed information systems is described in this chapter and a number of prevalent systems are reviewed.

5.2 Distributed Information

Historically, mainframe computers provided centralised storage and processing services to communities of users, but over the last two decades we have witnessed a shift towards personal computing. As the price of computer equipment fell, increasingly powerful personal computers became popular. A consequence of this *down-sizing* has been that processing and storage shifted from the mainframes to networks of workstations and personal computers. This move has had a significant effect on the manner in which digital information is organised, accessed and manipulated.

Personal computers have popularised computing and encouraged enormous growth. In some respects personal computers have liberated the user, allowing a greater degree of autonomy over the way in which they operate rather than having to conform to the policies of a centralised system. As a result of this phenomenon, many smaller centralised computers now exist, each providing local services and connected to other computers via a network. It is possible to see how all the information once stored on a mainframe can now be distributed across a number of personal computers.

5.3 Distributed Resource Discovery Systems

As information is now likely to be distributed across machines, questions arise as to whether information can be shared, located and accessed by remote users. The File Transfer Protocol (FTP) [21] provides a means of navigating a remote file system and retrieving documents over a network. However, this can only be achieved once the user has determined the network address of the FTP server which hosts the file and its location in the file system hierarchy. This can be a non-trivial task.

A number of solutions have been developed over recent years which provide tools for accomplishing these tasks. These distributed information systems allow users to make information accessible from servers running within their network domain. This allows users connected to the same network, regardless of their geographical location, to access the information on these servers via client programs.

This genre of distributed information system usually adheres to the client/server model. This model allows a high degree of device independence as the client and the server programs need only be able to interpret the appropriate protocol suite and data formats to offer a heterogeneous, distributed information system. This approach allows the freedom to develop custom user interfaces for client programs.

Once the province of select computing departments and academia, the Internet has expanded at a furious rate over recent years. The rapid growth of the Internet can be observed in figures 5.1 and 5.2, which show the rise in the number of computers connected and the number of network domains respectively. It has been estimated by Network Wizards[1] that by the start of the new millennium the number of Internet hosts will be in excess of 100 million.

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

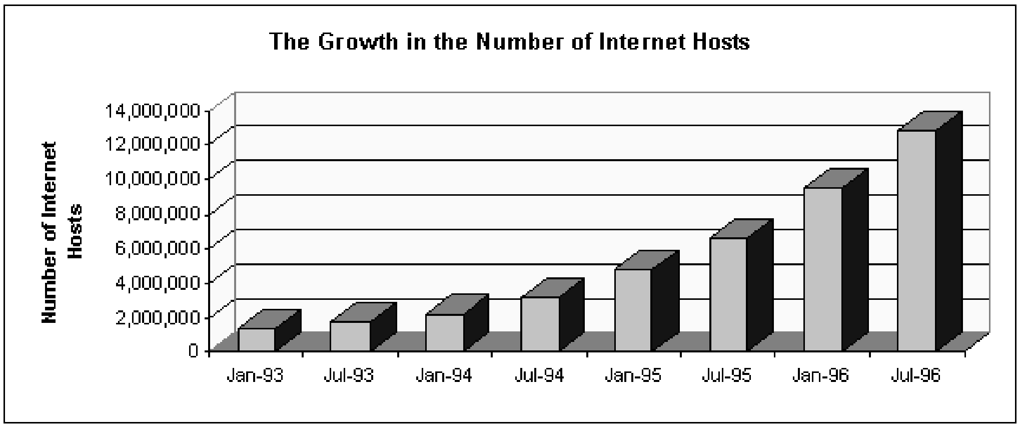


Figure 5.1: The rapid growth in the number of Internet hosts. Source: Network Wizards (http://www.nw.com)

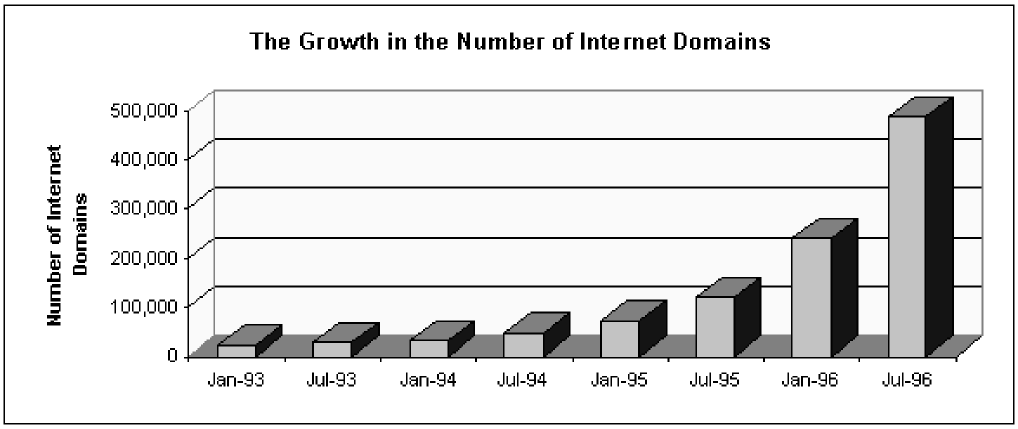


Figure 5.2: The rapid growth in the number of Internet domains. Source: Network Wizards (http://www.nw.com)

It is clear that as the Internet continues to grow then, so too, will the amount of information available across it. This conclusion has stimulated much interest in the provision of distributed information services, in particular resource discovery systems. A survey of the most popular resource discovery systems can be found in the following sections.

5.3.1 Archie

As the Internet has expanded, a commensurate rise in the number of FTP servers has also been observed. As such it can be an arduous task to locate files residing on remote FTP servers whose Internet addresses are not known in advance. Archie [63] is a tool that aids the user in searching for a file in Internet FTP archives.

Archie is a database service that indexes files that are publicly accessible from FTP servers on the Internet. An Archie server processes a query search string from a client and returns a list of results that matched against its indexes detailing the location on the FTP servers where each file resides.

The index and search facility provided by Archie greatly reduces the effort required to locate information on the Internet, but the name of the file required needs to be known in advance by the user. The database system maintained by an Archie server is not truly distributed, as each server co-ordinates itself via a master site. The master site generates a master database by interrogating registered FTP servers at periodic intervals.

5.3.2 Wide Area Information Server (WAIS)

The Wide Area Information Server (WAIS) [120] was developed by Thinking Machines corporation. A WAIS server provides access to a list of information sources, with each source providing access to a specific type of information. A user can select appropriate sources and nominate keywords which are then combined to form a query and subsequently disseminated amongst WAIS servers. The servers supporting the selected sources compare the user's query against a full text index of the available documents and respond with a list of matches. A ranking between 1 and 1000 is prepended to each document returned in the list of matches indicating its potential relevance. Relevant documents can also be grouped for use in searches to further refine the collection of documents.

A server exists which acts as a directory of all registered WAIS servers that provide information regarding certain sources. This can be queried by WAIS users and can provide a valuable reference point prior to beginning a search.

Although the majority of information available through WAIS servers is of a textual nature, as it is less complex to index, it is possible to make multimedia information available through the use of simple keyword associations.

5.3.3 Gopher

The Gopher system [3], developed at the University of Minnesota, is modelled on that of a distributed filing system. The information residing on a Gopher server is structured as a hierarchy in which intermediate nodes are equivalent to standard directory listings and leaf nodes are regular documents. Additional attributes are encoded within the Gopher hierarchy to indicate the media types of the nodes present. This information helps the system when attempting to render the documents.

On a superficial level, Gopher can be appreciated as a more versatile alternative to FTP. The strength of the system lies in its ability to incorporate the hierarchies of remote Gopher servers transparently and present a seamless view on a distributed document space, or *Gopherspace*. This approach allows a distributed, but logical, hierarchy to be constructed with pointers to remote Gopher servers allowable at any location in the tree. As with WAIS, Gopher search servers can be used to generate a virtual node as the result of a search applied across a bounded region of Gopherspace.

The interactive, browsing nature promoted by Gopher has encouraged users to make available a significant proportion of multimedia information through the system. A new version of the Gopher protocol, called Gopher+ [2], has been proposed to enhance the support for non-text media. Although Gopher attracted a considerable amount of attention, it appears to have been usurped by the World Wide Web.

5.3.4 Harvest

A more sophisticated resource discovery tool than those described above is the Harvest system [28], developed at the University of Colorado. Harvest provides an integrated set of customisable tools for gathering information from diverse repositories, building topic-specific content indexes, flexibly searching the indexes, widely replicating them, and caching objects as they are retrieved across the Internet. The system is able to interoperate with WWW clients and with HTTP, FTP, Gopher and Usenet information resources. It aims to reduce server load, network traffic, and space requirements when building indexes, compared with previous systems.

5.4 The Vision of a Global Information System

A complementary approach to the searching techniques described above, is to provide mechanisms for allowing semantic associations between ”chunks” of information to be created and explored. The concept of automated non-linear text was first described in 1945 by Vannevar Bush [35] through a memory extension device called “Memex”, in which an individual could store books, records and letters in a form such that specific information could be rapidly retrieved. It was an ambitious scheme for a large-scale publishing system, with connections or *trails* between related documents, which could be created by any user of the system. Information was to be stored on microfilm, and could incorporate both textual documents and pictures. Bush acknowledged that some advances in technology would be necessary to bring the scheme into existence, but irrespective of this he received considerable support for his ideas.

In the 1960s, Engelbart worked on a computer system which acted as a reasoning and research tool for assisting the intelligence of its users. This system became known as NLS (oN Line System) or Augment [64], and was probably the first working system to incorporate hypertext ideas successfully into its design. Conceived and designed as a tool to aid structured thinking, NLS allowed for a hierarchical mix of text and images to be made and linked together. The commercial version was aimed at the office-automation market and included sophisticated communications facilities with implicit hypertext linking, such as electronic mail, bulletin boards and teleconferencing, together with a variety of office tools.

Theodore Nelson used the term hypertext in the early 1960s to describe the idea of non-linear reading and writing implemented on a computer system for annotating and connecting associated ideas. The Xanadu project [144], was the vehicle through which he experimented with these notions. Xanadu was a fusion between Nelson’s own ideas and those from Bush’s Memex system and Engelbart’s NLS/Augment; however Nelson’s ideas and plans for the project encompassed not only the technical details of the system, but also the future social implications. He attempted to anticipate the effect that the information revolution would have on society, and predicted a future in which Xanadu would play a key role. Xanadu is an ambitious design for a large-scale, distributed hypermedia system. The design incorporates linking facilities, information publishing, a system of royalties for authors of material and accounting facilities which aimed at making Xanadu an integral aspect of modern-day society.

5.5 Hypermedia Systems with Local Distribution

Early popular hypermedia products, such as Guide [29] and Storyspace [61], are single user/single machine systems aimed at the personal computer market. At the opposite end of the spectrum are a small number of hypermedia systems that were developed from the outset with intrinsic support for distribution over Wide Area Networks (WAN), such as the Internet. These systems are reviewed in section 5.6. Between these two extremes lie a number of hypermedia systems that are able to operate in a distributed environment but possess no inherent capability for distribution. Hypermedia systems that fit within this category can achieve limited distribution through the services provided by the host environment. By virtue of operating within a UNIX environment, systems running on workstations can take advantage of facilities such as Network File Systems (NFS) and network windowing systems (X-Windows). Many hypermedia systems within this category also rely upon a single, central, commercial database which is used by clients as a hypermedia object server. A selection of systems that are configured in this manner and able to function within a Local Area Network (LAN) are reviewed in the following sections.

5.5.1 Neptune/HAM

Neptune [57] is a hypertext system developed at Tektronix' Computer Research Laboratories that was conceived to support Computer Aided Design (CAD). The abstract machine used by Neptune was later formalised as a generalised hypertext engine called the Hypertext Abstract Machine (HAM) [36]. HAM is a general-purpose, transaction-based, multi-user server for a hypertext storage system. As such, it provides a range of basic services for the storage and manipulation of, not only nodes and links, but also hypertext graphs and contexts for data partitioning. The authors illustrated the wide applicability of HAM by modelling the hypertext structures of a diverse range of hypertext systems such as Guide, Intermedia and NoteCards. Multiple clients accessing a centralised server was a common model for distribution among the earlier implementations of hypermedia systems, with Neptune/HAM being no exception.

5.5.2 Knowledge Management System (KMS)

Out of the Zog Project, developed at Carnegie-Mellon University in the late seventies and early eighties, came the Knowledge Management System (KMS) [1]. KMS is a commercial hypermedia system which can support collaboration over a range of activities including on-line documentation, project management, software engineering and computer aided instruction. The entire system is geared towards supporting the rapid creation of and traversal between *frames* of information.

Akscyn *et al* believed that the computing technology of the nineties would comprise networks of large-screened, diskless workstations. The design of KMS was optimised to exploit their view of future computer architectures. A single master server holds details of all storage devices containing KMS frames. It is now commonplace for workstations to have a significant amount of storage space, and although KMS can operate in a distributed environment, the design of the system does not map efficiently onto contemporary technology.

5.5.3 Intermedia

Intermedia [96] from Brown University was an influential distributed hypermedia system. The architecture, described in some depth in section 3.3, was such that all Intermedia clients were configured to access a single central hypermedia database server. The design of the system proved limiting in that not only were Intermedia clients required to access a single hypermedia server, but in addition there was no mechanism to facilitate communication between servers. These restrictions inhibited Intermedia clients from incorporating resources external to the designated hypermedia server. By relying upon a single hypermedia server the operation of Intermedia is effectively restricted to a local area network with no inherent means of scaling the architecture further.

5.5.4 Sun Link Service

Introduced earlier in section 3.4, the Sun Link Service [154] was the first example of an *open* distributed hypermedia system. Pearl advocates a loose coupling between the management of nodes and the management of links. Any modification and delivery of the nodes is managed by independent applications, leaving the linking functionality to be supplied by a shared back-end process. Although the Sun Link Service was specifically developed to operate in a distributed environment of Sun workstations, Pearl did not address the distribution of link information beyond a single server. As such there is no mention of a mechanism by which link servers can communicate to share additional information.

5.6 Hypermedia Systems with Global Distribution

The design of systems such as Neptune/HAM and Zog/KMS were all heavily influenced by the contemporary computer hardware available. Although these can be classified as distributed hypermedia systems their design and reliance upon a single, central hypermedia database server meant that their deployment only proved feasible on a small scale. It is now commonplace to see heterogeneous networks of workstations and personal computers, most with some form of local storage, all having access to designated servers with additional resources. Modern systems should be designed with the flexibility to exploit such resources.

Little consideration was paid to the extension of early hypermedia systems beyond that of their solitary local server. Without these extensions Van Dam [177] recognised that as users were conditioned to operate in isolation such systems were reduced to *information islands*, unable to enrich the user's experience with resources from remote systems. Although still based upon the client-server model, the more recent generation of open hypermedia systems, such as HyperDisco and DHM, allow their clients to connect to one or more of their local hypermedia servers. This additional measure benefits the user in that a greater wealth of material can now be made available to them.

As mentioned in section 5.3, the rapid growth of the Internet and a corresponding rise in the amount of electronic information available on-line has been witnessed. Over recent years systems have been developed to address the challenges of large-scale management, integration and delivery of hyper-linked information over WANs. In the following sections the most influential distributed hypermedia systems are reviewed, each capable of delivering hypermedia functionality over the Internet.

5.6.1 World Wide Web (WWW)

The World Wide Web was introduced in 3.2, in which various aspects were described but without specifically alluding to the distributed nature of the system. In common with many distributed information systems, the WWW adopts the familiar architecture of multiple servers and heterogeneous clients. The simple, yet highly effective, design of the WWW allows almost transparent access to documents anywhere over the Internet. This ability, coupled with appealing content presentation and basic hypertext functionality, has led to the general acceptance and widespread application of the WWW.

As explained earlier, the WWW uses a proprietary document format called HTML [17] for conveying hyper-linked information. The WWW also supports the transfer between client and server of any MIME[2] encoded document, regardless of whether the client has the ability to render the document.

WWW clients communicate with WWW servers using the Hypertext Transfer Protocol (HTTP) [15]. In addition to HTTP, the more advanced WWW clients support an increasing number of the common Internet protocols such as FTP [21], Gopher [3], POP [141] for electronic mail and NNTP [122] for Usenet news. This ability allows WWW clients to provide a single, consistent interface to a variety of distributed information sources.

Any document accessible via the World Wide Web is referenced by Uniform Resource Locator (URL) [18]. A URL is a text string that comprises the Internet protocol, or access type; the Internet address of the host on which the document resides, and optionally the port number to use; the file path of the document to be retrieved. The generic format of a URL is as follows:

```
<access type>://<Internet host address>[:port]/<document path>
```

The path component of the URL is formatted according to the type of service being accessed. By using reserved characters, a URL can denote an anchor in an HTML document, or contain parameters to a CGI script. WWW hypertext links to destina-

[2] Multipurpose Internet Mail Extensions

tion documents are also specified by URLs, and, through the suite of Internet access protocols, links to any information type can be supported transparently to the user.

The URL provides a succinct and generic format for referencing any document accessible across the Internet, which has led to it becoming rapidly accepted as a *de facto* standard. URLs are undoubtedly convenient due to their visible and portable nature, but can also be regarded as volatile because such a naming scheme is entirely location dependent. A more detailed discussion of this topic can be found in section 7.9.

The WWW allows the user the option of employing two additional techniques to seek an increase in performance. One technique is the use of a client cache in which copies of the most recently visited documents are stored. When the user revisits a document that has been cached, the client contacts the original server to compare the time stamp of the cached document with the version stored there. If the time stamps differ a copy of the updated document is requested, otherwise the client displays the cached version.

Document caching at the client only benefits a single user, but multiple users can benefit by employing a proxy server. A proxy server is a separate process that can masquerade as both a WWW client and a WWW server. It is by virtue of this duplicitous role that a proxy server is able to act as a remote cache. WWW clients can be configured to contact a specific proxy server to retrieve a document rather than contacting the original WWW server. If a requested document happens to be cached at the proxy server then a copy is sent to the client. If this is not the case, the proxy server obtains a copy from the original WWW server and forwards a copy to the client. A performance increase can usually be observed by using a popular local proxy server.

5.6.2 Hyper-G

Previously introduced in section 3.8, Hyper-G is a large-scale, distributed, open hypermedia system. The system is decomposed into a UNIX based server and a client, available for both UNIX and Microsoft Windows. Unlike the WWW, Hyper-G preserves document sanctity by storing the links separately.

Each information provider has a local configuration comprising both a link server

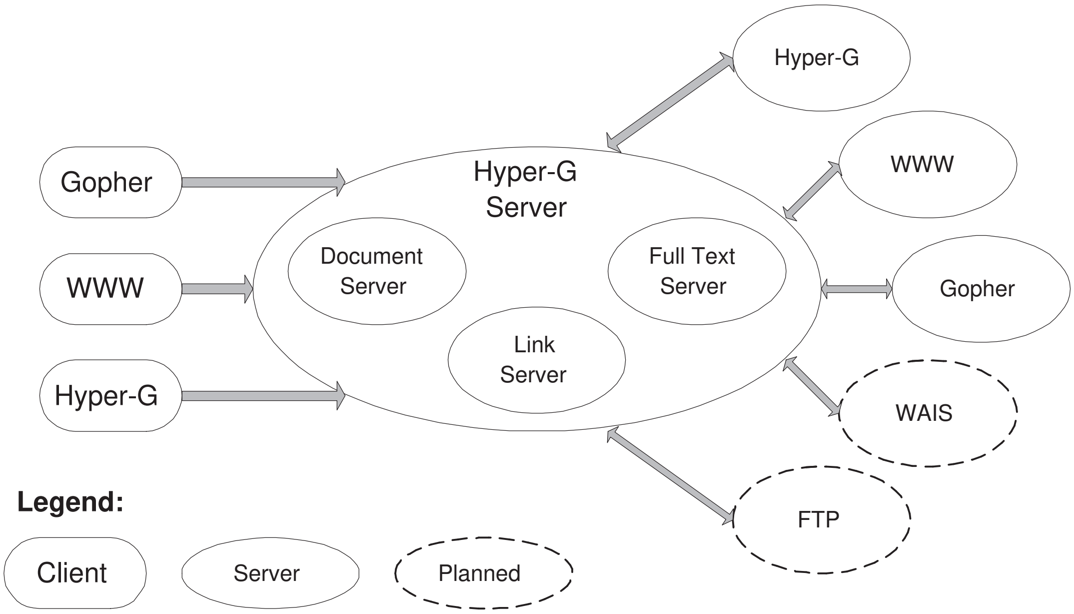


Figure 5.3: The Hyper-G architecture.

---

and a document server. Before a presentation client may render a document it must first issue a request to the link server supplying the document object's unique identifier. The link server responds with the document's respective access details and any pertinent anchor information. The client then uses the access information to retrieve the document. These concepts were extended to accommodate the Internet. All unique object identifiers were prepended with their respective server credentials. A central registry issues Hyper-G server identifiers based on Internet domain names. The document server was also extended to cache frequently accessed objects and reduce network traffic and waiting time.

The team at Graz have managed to achieve a high level of integration of external information systems with Hyper-G. As well as the proprietary Harmony and Amadeus clients, users may also interact with a Hyper-G server through a WWW browser. Although such users will obviously receive a poorer quality experience due to the lack of client side tools, this approach nevertheless allows a much wider audience to use the Hyper-G link services. In addition to creating links between Hyper-G documents, users may also link to WWW and Gopher documents as these protocols are also supported by the Hyper-G server.

In section 3.12, the issue of maintenance was raised when contrasting the effort required to manage a Hyper-G server and a WWW server. By holding the links separately from the documents and replacing absolute URLs with a relative naming scheme, the task of managing a large volume of information can be eased. These arguments, in addition to users being able to access Hyper-G servers transparently from a WWW browser, lead Andrews *et al* [7] to conclude that Hyper-G servers are better placed to serve information to the WWW.

5.6.3 Virtual Notebook System (VNS)

The aim of the Virtual Notebook System (VNS) [166], developed at the University of Houston in Texas, was to enhance the productivity of medical research scientists. VNS was designed to facilitate information acquisition, sharing and management in a collaborative working setting. To support collaborative working it was essential for VNS to operate in a distributed, heterogeneous computing environment.

Although VNS separates the links from the documents they are both stored centrally in a commercial relational database. Links are defined upon the particular user's view, so for a shared document the links available may vary between users. When a document is viewed the presentation application constructs the view from objects in the database. The original architecture required dedicated applications to build a presentation composed of objects from the database, but subsequent enhancements by Burger [34] allow modified third party applications to integrate with VNS.

The distributed architecture is based upon that of a centralised workgroup server (WGS) which provides hypermedia services to multiple client applications. The WGS provides storage and management of local information through SyBase, the chosen database. By allowing workgroup servers to communicate wider collaboration between VNS users is supported. Integration with external information systems is also supported through the use of *gateways* in the workgroup server. VNS is able to operate in a distributed and heterogeneous environment by running under UNIX and adhering to the X-Windows and TCP/IP protocols.

5.6.4 Distributed Hypertext (DHT)

The Distributed HyperText system (DHT) [147], developed at the University of Southern California, provides a flexible infrastructure within which a diverse collection of information services can be incorporated. DHT does not provide a central hypermedia server, but instead, defines an open protocol for information access which server processes can obey. As such, DHT provides a standard and consistent interface through which client processes can use the services offered by the utilities integrated within the suite. Each utility within the suite has a dedicated server process which acts as a gateway, translating between the DHT protocol and the protocol associated with the service.

The design of DHT has the advantage that uniform access can be achieved across any integrated service. There is no requirement for the information to be translated to any specific format and it can be readily used in its native form. Each new server process is required to conform to the common interface and must be developed and configured accordingly. One disadvantage of this approach to integration is that a significant proportion of functionality is duplicated within each server process.

5.6.5 Multicard

Introduced earlier in section 3.5, Multicard has undergone significant architectural restructuring in an effort to extend its applicability to a WAN [158]. There are five principle components to the new design: a user station, a kiosk, an application server, a content server and a link server.

The user station typically consists of a GUI with presentation/editing resources and network connectivity. A MHEG engine is responsible for the interpretation and execution of the MHEG objects. The MHEG engine resolves the object's internal links and relies upon the link server to resolve external links.

From the end-user perspective the kiosk is the service entry point. The kiosk service is the only address known by the user station. The main function of the service entry point is to connect the user requesting a service to the application server that offers the service.

An application server defines application service elements and must register its own services with the link server. Each service element interface is coded in a MHEG

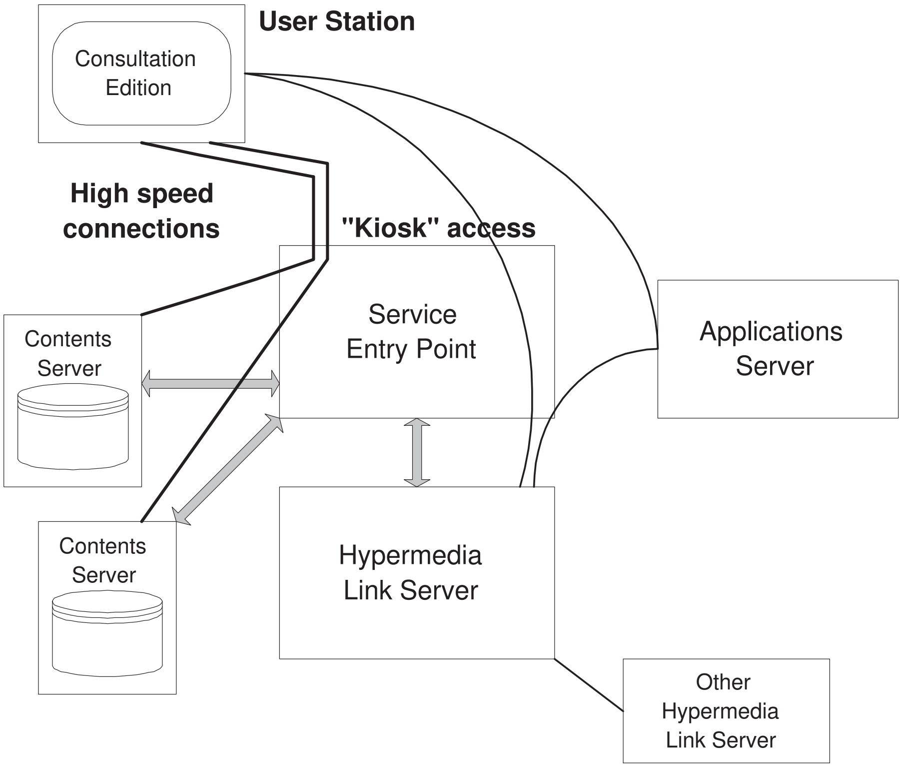


Figure 5.4: The Multicard architecture.

---

representation by the application server. An application server uses the link server facilities to store and access distributed data or to call services from other application servers. By using the hypermedia facilities of the link server, each application service element develops its own view of data and delivers an added value on this data.

The content servers are responsible for the multimedia data management. Two types of dialogue are required here. The first allows the kiosk to request data management facilities, while the second expedites the transfer of data to the user station.

The link server provides a set of hypermedia primitives. The server stores the hypermedia objects with their definition attributes, links and groups of nodes in a hypergraph.

5.6.6 Microcosm

The design of Microcosm was outlined in chapter 4, but the extensions necessary to allow the system to operate in a distributed environment are presented by Hill [109, 110]. The current model is enhanced, permitting users to *publish* any of the

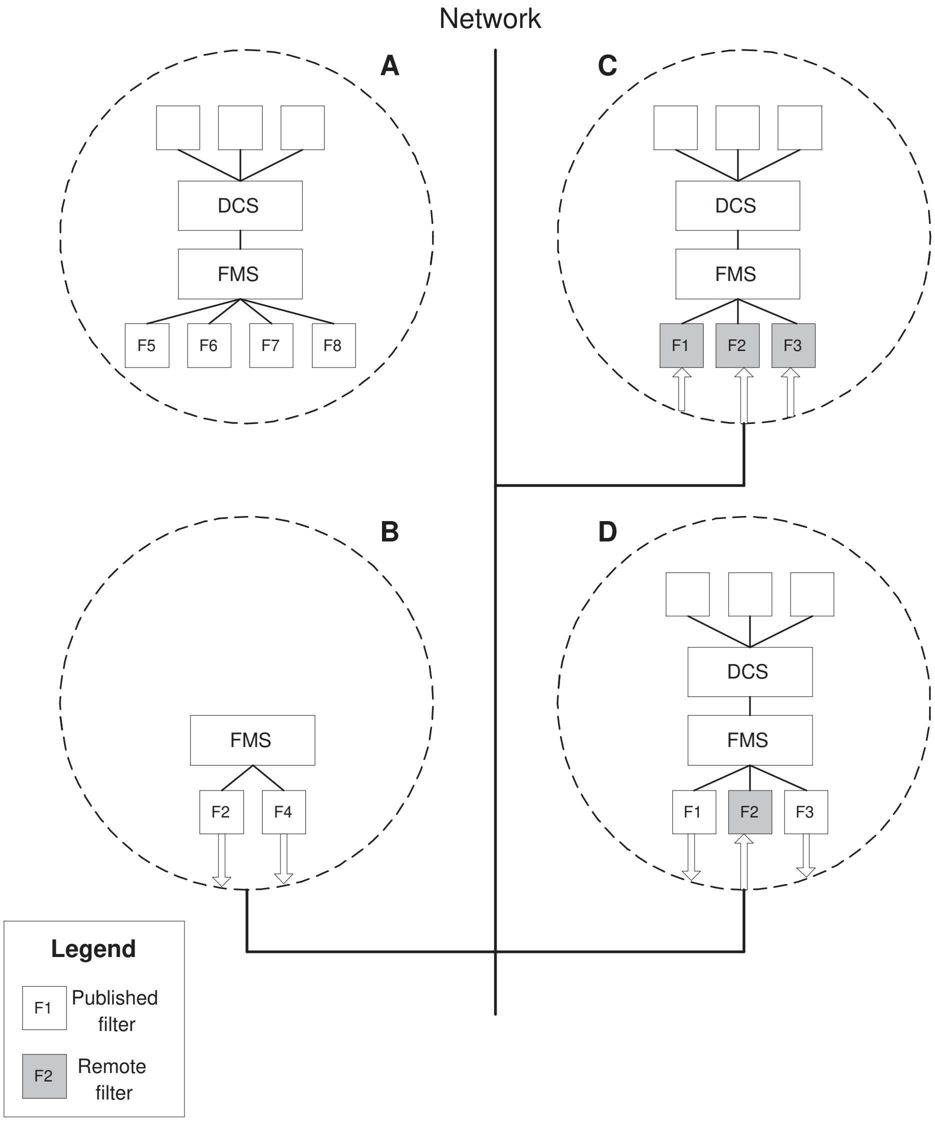


Figure 5.5: The range of Microcosm configurations supported by Hill.

---

filters within their Microcosm application session. This process effectively makes the functionality of these filters available, allowing remote users to connect to the session and include one or more of the published filters within their local configuration.

The majority of distributed information systems are designed with a clear client and server distinction, whereas the distributed model proposed by Hill allows for a greater degree of flexibility in the configuration. The four alternative configurations can be appreciated in figure 5.5. The first circle (A) depicts a stand alone user session not connected to the network; the circle sphere (B) shows a server session with published filters; the third circle (C) shows a client session using the remote filters from both (B) and (D); the fourth circle (D) shows a mixed client and server session. A rudimentary location service is provided allowing remote instances of Microcosm to be interrogated for a list of currently published filters.

Microcosm was extended in two central modules. The Filter Management System was adapted, enabling it to communicate with remote instances of Microcosm over the Internet via TCP/IP sockets. Additional information needed to be encoded within each message to enable communicating FMSs to route messages directly to specific published filters. The Document Control System was also extended to facilitate the retrieval of documents from a remote host. The format of the unique document identifiers associated with every document under the control of the DCS was extended to adopt the style of the URL used by the WWW. The access mechanism denoted in the URL allows the DCS to accommodate information resources obtained through services such as WWW, Gopher and FTP.

5.6.7 HyperDisco

Discussed earlier in section 3.11, HyperDisco [183] was implemented by using the Hyperform hypermedia system development environment [181]. Recent research by Wiil *et al* [184] involved extending the HyperDisco infrastructure to support Internet distribution.

The concept of a logical *workspace* is introduced as a Hyperbase Management System (HBMS) that acts as a gateway to a set of multimedia files. Workspaces offer a fixed set of hypermedia services, from file content retrieval to version control. It is possible to configure workspaces to permit private, public or group entry. Group and public workspaces allow users to rendezvous and collaborate as hypermedia authors.

To facilitate Internet support, HyperDisco has been extended with `get-file` and `put-file` operations for location transparency. A name server is also supplied in which a user must enter hostname and port information for all public and remote workspaces that they wish to access.

5.7 Distributed Link Services Above The WWW

In addition to developing bespoke architectures for open hypermedia systems, a number of recent projects are investigating using the WWW as an infrastructure for providing a distributed link service over the Internet. A number of these projects are described in this section.

5.7.1 Distributed Link Service

The Distributed Link Service (DLS) [39] is an initiative from the University of Southampton in which the application of an open hypermedia link service to the World Wide Web is being investigated. Through the DLS technology, Carr *et al* have demonstrated that it is possible to embody the Microcosm philosophy of open linking above the WWW to provide added value. In the same way that a client can connect to a remote Web server to access a document, the DLS allows a client to connect to a link server to request a set of links to apply to the data in a document.

The DLS implementation consists of a client-side interface tool and a set of server-side link service CGI scripts. The interface tool provides a mechanism for selecting a particular link database, and also for allowing the user to nominate a particular action to apply to their selection. With minimal configuration, users of the WWW can benefit from the advantages of an open link service in an almost transparent fashion.

5.7.2 Aquarelle

Aquarelle [160] is a European project that aims to provide access to information about European cultural heritage including sculptures, artwork and monuments. *Primary data* is produced by the cultural organisations which reside on archive servers, and *secondary data* is specialist information regarding specific collections which is held on structured folder servers. Aquarelle provides access to the resources via both information retrieval by querying and through hypertext navigation.

At the core of Aquarelle lies an access server responsible for providing a distributed link service and connecting users to archive and folder servers. The link service will support the creation of links by third parties. It is expected that professional Aquarelle users will be pay for the services and, as such, a key requirement is to maintain a high degree of link integrity. As the success of the system relies upon the vigilance of content providers in keeping links to their material up to date, Aquarelle is founded upon an economic model whereby content providers pay a registration fee and are remunerated for access to their resources.

5.7.3 DeVise Hypermedia (DHM)

As described in section 3.9, a hypermedia development framework and prototype applications have been developed within the DeVise project [90, 91]. Grønbæk *et al* also selected the WWW infrastructure as a vehicle for delivering their Dexter-based link services across the Internet [87].

The general architecture of DHM/WWW has a Java applet responsible for browser integration and communication with the Hypermedia Service. The applet provides an interface to the Hypermedia Service functionality and supports the DHM protocol. All anchors returned from the link service are encapsulated in Javascript and inserted into the HTML causing the browser to call back the applet when a link is followed. Although this solution is platform independent, it is browser dependent as the implementation requires the use of specific Netscape technology to facilitate communication between Javascript and Java applets.

Two further platform dependent solutions were investigated as an alternative to the Java applet: the use of plug-ins and the Software Development Interface (SDI) under Netscape; the use of OLE automation under Explorer. It was concluded that the technology is not sufficient for a platform/browser independent solution.

5.7.4 Chimera

Chimera [6] is an open hypermedia system and was discussed in more depth in section 3.7. Anderson [5] describes three distinct experiments in which Chimera is integrated with the WWW.

A WWW server written in Java was extended to utilise the Java API of Chimera in the first experiment. This provides users with a lightweight mechanism for traversing Chimera hyperwebs. The second experiment concerns the development of a Chimera enhanced website. A number of inhibitors to the distribution of Chimera were overcome by redesigning the architecture and using URLs to access hyperwebs. The final experiment builds on the previous two by attaching a Java applet to each HTML page to provide ubiquitous access to Chimera services. A number of technical workarounds were required to achieve a satisfactory solution.

5.8 Summary

As the trend of downsizing continues, the number of networked computers with personal storage will grow. This phenomenon, in turn, engenders the further distribution of information and services. A number of resource discovery tools were reviewed in this chapter that can assist in locating and retrieving documents that may reside on an unknown host.

Although many hypermedia systems are implemented as single user/single machine products, a range of hypermedia systems with local distribution were reviewed in this chapter. Neptune/HAM, KMS/Zog and Intermedia are examples of systems that possess little inherent capability for distribution but are able to operate in a distributed environment by taking advantage of native services such as NFS and the X-Window system. While many of the systems in this category rely upon a single, central hypermedia database server, DHM and HyperDisco are examples of modern open systems which allow access to multiple servers simultaneously.

It is interesting to note that when Frank Halasz delivered his 1991 keynote address "Seven Issues for the Next Generation of Hypermedia Systems" [97], issues specific to the design of globally distributed hypermedia systems did not feature among his list. It is possible though that this was implicit through his plea for further research into computer supported collaborative working. Van Dam [177] was acutely aware that the hypermedia community were building *docu-islands* using disparate systems which were unable to communicate and exchange information with each other.

The designers of VNS were the first to address seriously the issue of widespread distribution and access to information by developing a system in which the *work-group servers* communicated. While VNS demonstrated some success in this respect, the non-extensible design of the system allows minimal scope for alternative linking strategies. The designers of DHT largely achieved their aim of integrating a range of disparate systems with a hypermedia front end by using a suite of protocol converters to create a unified interface. This approach however, subsequently renders the original functionality of each of these systems inaccessible to the user. Additional effort is also required to produce a specialised gateway for each system to be integrated.

Neither Halasz nor Van Dam could have predicted the impact that the World Wide Web would have upon the Internet community. The essence of the WWW is captured in its ability to provide transparent access to documents anywhere on the Internet via hypermedia links using a simple, but effective, architecture. Although open in terms of the protocols supported, the WWW is classified as a closed hypermedia system with respect to the Flag Taxonomy [150]. In addition to serving its proprietary clients, Hyper-G is also capable of acting as both a document manager and link service to WWW clients.

The new design of Multicard represents a significant departure from the original. The system allows an application server to access multiple content and link servers before delivering the information to the user station, but whether MHEG objects can be used in a sufficiently open manner remains to be seen. Unlike Hyper-G, Microcosm can offer an array of link types and a set of link services that can be dynamically extended on a per-user basis. While the versatility of the distributed model of Microcosm is not in question, a critique of the Filter Management System in chapter 6.3.1 raises doubts as to the suitability of such a topology for the basis of a distributed system.

The DLS, Aquarelle, DHM and Chimera propose alternative strategies for integrating with the WWW to rectify many of the current shortcomings through the provision of a link service. One of the guiding principles of open hypermedia is for the seamless integration of desktop applications. While the attractions of providing an open link service above the WWW are clear, we must guard against integrating with the WWW at the expense of desktop applications.

Many of the systems described in this chapter have demonstrated, albeit with varying degrees of success, how distributed hypermedia can be delivered. The remainder of the thesis is dedicated to describing the design of Microcosm: The Next Generation (TNG). Microcosm TNG is a distributed, open hypermedia system with a design that represents a significant departure from the original Microcosm architecture. Distributed systems have an inherent reliance upon an underlying communication substrate. The next chapter analyses the existing Microcosm communication models, reviews contemporary message passing solutions and describes the new communication subsystem developed for Microcosm TNG.

6 The Design of an Open and Customisable Distributed Communication Layer

6.1 Introduction

As described in chapter 4, the original version of Microcosm executes on a single machine. Experience has shown that when Microcosm is applied to large-scale commercial applications, such as Pirelli [44, 45] and Trant [60], the performance of the system slows considerably. One solution to this problem is to sub-divide the responsibilities of the system and distribute them across a network of machines, therefore spreading the processing load. In chapter 5 various systems for managing distributed information were reviewed. While the architects of distributed hypermedia systems address the issues of providing and retrieving *distributed information*, they have often neglected to design systems with the inherent capability for the *distributed processing* of this information. In this chapter the existing communication models for Microcosm are analysed, contemporary message passing solutions are reviewed and the design of a distributed communication subsystem for Microcosm TNG is described.

6.2 Issues in Distribution

The movement from the traditional single processor application architecture to a client-server architecture operating over a heterogeneous network can yield a greater efficiency. This transition can also be improved upon by specifically designing applications in a distributed manner so as to utilise the resources available in an optimum fashion. Developments in technology have seen enterprises moving away from monolithic applications with centralised resources to suites of open systems workstations clustered in highly connected networks. The motivation behind decentralisation is largely economic. Tanenbaum [172] asserts:

> ...the most cost-effective solution is frequently to harness a number of cheap CPU's together in a system. Thus the leading reason for the trend toward distributed systems is that they potentially have a much better price/performance ratio than a single centralised system. In effect, a distributed system gives more bang for the buck.

6.2.1 The Advantages of Distributed Systems

In addition to the economic argument, a distributed system has the potential to have more total computing power than a centralised machine. The loosely coupled topology of a distributed system can facilitate incremental growth, allowing more computing power to be added later. Distributed systems also have the potential to be more reliable. If one machine crashes, then the rest of the system may still survive. Some applications are inherently distributed as both the data and machines are geographically separated. The ability to spread the workload over a number of machines can provide a more efficient and flexible system.

6.2.2 The Disadvantages of Distributed Systems

Distributed systems have an inherent dependence on network reliability and performance. Much research has focused upon network technology, but unfortunately network failures remain a consideration for all distributed system designers. Although distributed systems can provide convenient mechanisms for shared access to data, this ability can also pose an associated security risk.

6.2.3 Design Issues

Summarised below are a number of key issues that are considered to be of importance in the design of an open distributed system.

Transparency It is likely that a distributed system will manage processes and data that are physically located in different places. It is important to strive toward *location transparency*, as much as is sensible to do so, and present the user with a unified view of the system.

Openness An open protocol can promote greater integration and interoperability between participating entities. In addition, the issue of heterogeneity is of equal importance within a networked environment.

Flexibility Allowing the systems to be configured in a variety of ways provides greater flexibility. Resources can then be allocated in the most appropriate manner.

Scalability A scalable architecture can withstand additional users, machines, processes and data with minimal impact upon the overall performance.

6.2.4 The Importance of Communication

Modular architectures generally have the advantage that new components can easily be added. The flexibility of systems composed in this manner is realised through the communication and cooperation between the individual components. It is clear that the versatility of such systems owe much to the relative strengths of the communication layer.

6.3 Analysis of the Models of Communication for Microcosm

The following sections provide a brief analysis of the three existing non-distributed models of communication designed for Microcosm so far. A more detailed treatment of these issues can be found [75, 77]. This analysis fed into the design of a distributed communication model for Microcosm TNG described in section 6.6.

6.3.1 Segregated Communication Model (SCM)

This model supports two classes of process, viewer processes and filter processes. The Document Control System (DCS) controls the viewers while the Filter Management System (FMS) co-ordinates a serial chain of filters, see figure 6.1. All inter-process communication is routed via one of these two modules, and in some instances both.

6.3.1.1 Process Communication

The distinction between viewer and filter processes gives rise to a combination of four possible communication pathways:

viewer to filter When a viewer needs to communicate with a filter it posts a message to the DCS module. This message is then routed to the FMS, which ensures that it is propagated through the filter chain where it may be intercepted by one or more filters.

filter to viewer When a filter needs to communicate with a viewer it posts a message to the FMS. The FMS routes the message through the remaining filters in the chain and then on to the DCS. Finally, the DCS posts the message to the pertinent viewer.

filter to filter When one filter wishes to communicate with another it posts a message to the FMS, which as mentioned before, routes the message through the chain where it may be intercepted by one or more filters positioned further along the chain.

viewer to viewer If one viewer wished to communicate with another a message would have to pass first to the DCS, on to the FMS, through the entire filter chain before returning to the DCS, where it would eventually reach the desired viewer.

6.3.1.2 Scalability of the SCM

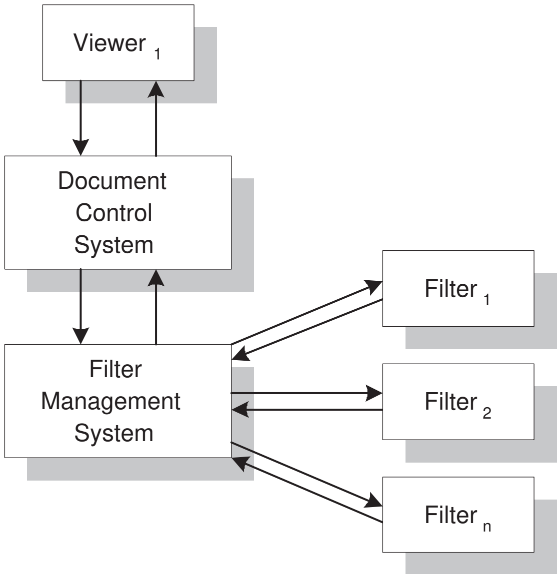


Figure 6.1: The Segregated Communication Model.

The SCM possesses two features that restrict both its performance and scalability. The first is the bottleneck between the DCS and the FMS, while the second is the serial, uni-directional nature of the filter chain. The majority of the communication overhead in the SCM is attributable to *all* messages being propagated through the entire length of the filter chain. Every filter must pass on every message in order that successive filters can receive them. This illustrates the deficiencies in supporting a concurrent system with a serial communication pathway.

$$2f + 3 \tag{6.1}$$

$$k \sum_{i=0}^{i=L} 2(f - (p + i)) + 3 \tag{6.2}$$

$$total = (6.1) + (6.2) \tag{6.3}$$

Formula 6.1 is the number of steps a message is routed through when a viewer issues a request for information regarding document links to a linkbase(s), where f is the number of filters present in the chain. Formula 6.2 represents the number of steps taken when L linkbases at consecutive positions $p + i$ in the filter chain respond with a mean of k links. Equation 6.3 is the summation of the previous formulae giving the total number of transitory stages messages must travel through for this common scenario.

The FMS is almost stateless, which has the advantage that the filter manager makes no assumptions regarding any of the filters in the chain. The major disadvantage of this approach is that one filter cannot easily determine the presence of other filters. The *Available Links* filter, under this communication model, needs to know when every linkbase in the chain has finished processing so that the link destination documents can be presented. This is achieved through the introduction of control messages. Each linkbase issues a message when processing commences and another upon completion. These messages are intercepted by the *Available Links* filter which increments and decrements a semaphore to monitor this activity. These additional control messages required to support the scenario described above are not reflected within the formulae.

6.3.1.3 Benefits of the SCM

The ability to dynamically reconfigure the majority of the system is of immense benefit to all users. The functionality of the system can be augmented or diminished at any stage by interacting with a single dialog box. Davis *et al* [52, 124] describe the ease with which third party applications can be integrated with Microcosm, thus granting access to varying levels of hypermedia functionality.

6.3.1.4 Limitations of the SCM

The ordered, serial nature of the chain architecture inhibits a filter from holding a bi-directional conversation with another filter. In an attempt to accomplish this a filter would need to exist in two places within the chain, the first instance positioned before the receiving filter and the second instance after it. This scenario is clearly undesirable.

Two viewers may require synchronisation control to provide a composite view of multimedia resources such as audio and video. Such viewers would demand rapid communication to achieve the simultaneity required for the presentation. Viewer to viewer communication within the SCM is obviously impractical considering the convoluted route that all messages must follow to support this feature.

The SCM has no concept of either a private communication channel between two processes or a broadcast mechanism. A private communication channel ensures that a message posted from a source process arrives at the intended destination process without being modified or deleted. This is impossible in the current model as each filter is required to inspect every message, hence having the opportunity to modify the message contents, before it is propagated throughout the remainder of the chain. A broadcast mechanism allows one process to post a message to all the other processes in the system, which can often be a useful feature, but unfortunately, difficult to achieve with the SCM.

The explicit requirement of each filter being responsible for propagating the message along the chain leaves the system vulnerable to a rogue filter. Such a filter has the power to halt the system completely by simply refusing to forward any messages, or by deleting every message received. Any system open to potential abuse in this manner is clearly far from ideal.

The serial nature of the filter chain dictates that each message is passed from filter to filter in a predetermined order. Filters have been developed that must reside at a specific position in the chain in order to function correctly. If a naive user incorrectly orders the filters in their chain the consequences are undocumented. At best, Microcosm would cease to operate correctly, while at worst the effect could be detrimental depending upon the filters present. To make a judgement upon the positioning of a given filter, a user must be aware of the service(s) provided by the filter, whether it is dependent upon any other filters to fulfil the service, and if so, the nature of the dependency. Alleviating any responsibility from the users for correctly ordering a suite of filters would prove advantageous.

6.3.2 The Active FMS

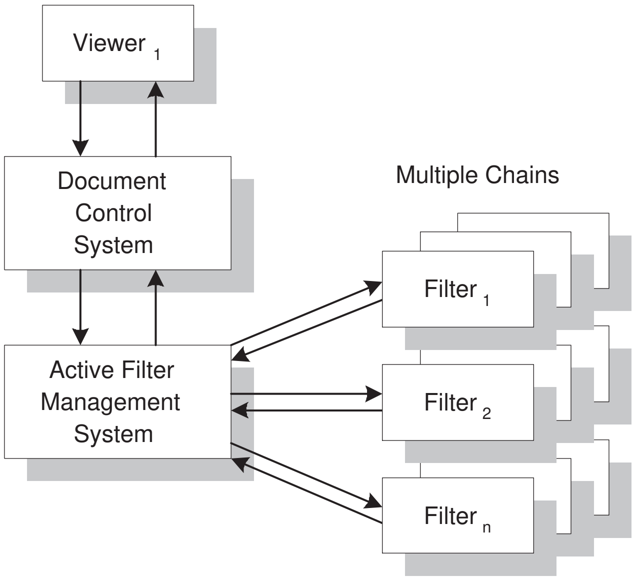


Figure 6.2: The Active Filter Management System.

The FMS has subsequently been improved upon by Hill [109], who recognised that while all of the benefits of the filter chain could still be retained, the efficiency could be improved. The optimisation of the FMS has been accomplished through the instantiation of a number of shorter chains, or sub-chains as shown in figure 6.2. Each sub-chain contains filters which have registered with the FMS to process the same *action*. Every new message sent is inspected first by the Active FMS to determine the action field, and then dispatched to the front of the relevant sub-chain containing filters that are registered to process this action. Through the eradication of superfluous messages in the filter chain, the net effect of the Active FMS is that every message is routed to the appropriate filter(s) in a smaller number of intermediate steps.

$$2L + 3 \tag{6.4}$$

$$k \sum_{i=0}^{i=L} 2(L - i) + 3 \tag{6.5}$$

$$total = (6.4) + (6.5) \tag{6.6}$$

Again, formula 6.4 is the number of steps a message is routed through when a viewer issues a request for information regarding document links to a linkbase(s), where L is the number of linkbases in the sub-chain. Formula 6.5 represents the number of steps taken when L linkbases at consecutive positions $L - i$ in the filter chain respond with a mean of k links. Equation 6.6 is the summation of the previous formulae giving the total number of transitory stages messages must travel.

The rise in performance afforded by the Active FMS is dependent upon the diversity of filters present in a given configuration. If a particular configuration specified only linkbase filters in the filter chain, only a single sub-chain would be constructed by the Active FMS. Although the Active FMS demonstrated a noticeable performance increase, the bottleneck between the DCS and FMS together with the limitations of the serial filter chain, remain as the central inhibitors of efficient communication.

6.3.3 The Direct Communication Model (DCM)

Wilkins *et al* [188] propose an alternative message passing strategy to the filter chain. The Direct Communication Model (DCM) abolishes the distinction between viewers and filters by promoting all aspects of the system to the process level. Processes are free to register and unregister with the DCM at any time. Once a process has registered with the DCM it is then able to *converse* with any other registered process in the system.

The DCM introduces the concept of a *topic* to denote the registration criteria for each process. For example, a linkbase would register an interest in receiving messages regarding the topic *follow link*. This would result in all messages concerning this topic to be delivered to all linkbase processes. For a process to register with the DCM a registration message must be supplied, typically containing the process name together with a list of topics. The subsequent dissemination of all messages by the DCM is based upon the registered topic(s) submitted by every process. The conceptual overview of the direct communication model is illustrated in figure 6.3.

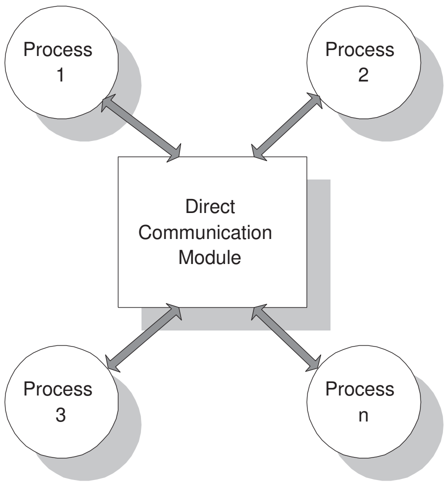


Figure 6.3: The Direct Communication Model.

6.3.3.1 Process Communication

A process can communicate with other processes via four different routes:

general broadcast a single process sending a message to all of the other processes.

process oriented one process sending a message to each process registered with the DCM under the nominated process name.

topic oriented one process sending a message to each process registered with the DCM under the nominated topic name.

process to process When one process sends a message to another process using a pre-allocated unique identifier.

6.3.3.2 Scalability of the Direct Communication Model

From figure 6.3 it is clear that the DCM does not suffer from the inherent latency present in the SCM and the Active FMS. Each message is sent from the originating process to the DCM, where it is then forwarded directly to the intended recipients. This sequence of events only ever results in two transitional steps between posting a message and its delivery, irrespective of the number of processes registered. The formulae and resultant equation presented below can be used to enumerate the intermediate steps travelled by messages under the DCM.

$$L + 1 \tag{6.7}$$

$$2Lk \tag{6.8}$$

$$total = (6.7) + (6.8) \tag{6.9}$$

Again, formula 6.7 is the number of steps a message is routed through when a viewer issues a request for information regarding document links to a linkbase(s), where L is the number of linkbases. Formula 6.8 represents the number of steps taken when L linkbases respond with a mean of k links. Equation 6.9 is the summation of the previous formulae giving the total number of transitory stages messages must travel.

The length of message pathways are considerably reduced, resulting in a linear equation regardless of the number of processes registered with the DCM. This crucial characteristic exemplifies that of a scalable communication architecture, whereas the efficiency of the two previous models is inextricably related to the number of filters in the chain.

6.3.3.3 Dynamic Behaviour

The registration entry for each process is held by the DCM, which provides a service allowing any process to query this information. Armed with this information each process can have complete autonomy over the destination(s) of any messages it may issue. The topology of the system now becomes completely dynamic, with the possibility of process registration entries changing constantly. The advantages that dynamic system behaviour brings are soured only by the slight increase in complexity required of each participating process.

6.3.3.4 Limitations of the DCM

The notion of ordering, and hence priority, is implicit within the serial filter chain. There is no provision within the DCM for messages to be received by processes on a basis of priority. A clash of interests would occur if two or more processes attempted to register to receive messages regarding a specific topic with the highest priority.

The SCM insists upon the FMS bouncing every message to each successive filter in the chain. Upon receipt of a message each filter inspects the content to discern whether it should be processed and/or forwarded. As a consequence of the implicit ordering of the filter chain, a filter has the capability to modify the contents of a message at this juncture prior to passing it on. The DCM, however, ensures that a message is delivered to each recipient process quasi-concurrently, allowing no opportunity for message modification.

6.3.4 Summary

The SCM enabled Microcosm to be developed into a very flexible system, allowing users to augment functionality and customise the configuration to suit their needs. The Pirelli project [44, 45] has demonstrated the limitations of the SCM when applied to a large-scale industrial application. The authored links were originally stored within twelve separate linkbases, allowing the links to be classified with respect to a subject domain. Grouping links into logical clusters has many advantages, for example, expediting ease of link maintenance. Unfortunately, due to the exceptionally high communication overhead incurred by having twelve separate linkbase processes in the filter chain, the links had to be rationalised into six linkbases.

The Active FMS eradicates the overhead previously incurred by the routing of superfluous messages to filters. The subsequent rise in performance is dependent upon the spectrum of filters included in a given filter chain. If a particular configuration specified only linkbase filters in the chain, only one sub-chain would be instantiated by the Active FMS.

Of the models of communication reviewed in this section, only the DCM boasts a scalable topology whilst retaining much of the flexibility enjoyed by the SCM and Active FMS. Only the DCM promises a consistent performance due to the efficient message routing.

Although the concerns regarding ordering and message modification are legitimate in theory, the requirements of typical Microcosm processes suggest that neither issue poses a significant problem. It is likely that scenarios can be formulated that are unable to be accommodated by the DCM, as equally there are scenarios for the SCM, but for the vast majority of configurations the advantages afforded by the DCM outweigh any theoretical reservations. As such the salient characteristics of the DCM have been adopted as the basis for the design of a new distributed communication model, described in section 6.6, which allows a single user session to be distributed across a number of machines.

6.4 Additional Communication Issues

Interprocess communication is often interpreted as involving only two parties, a client and a server. Sometimes there are circumstances in which communication involves multiple processes. Tanenbaum [172] identifies a number of different ways in which the internal structure of a group of processes can be organised.

Peer group is where all processes are considered equal.

Hierarchy group is where one process is the co-ordinator.

Overlapping is where a process can belong to multiple groups.

Group communication allows one logical connection to be used for sending the same message to multiple receivers, thus reducing processing at the sender. A flexible distributed communication layer should be able to model these three organisational structures. For this to be achieved, a mechanism for group addressing must be in place and correct message ordering must be guaranteed.

6.5 Review of Existing Distributed Technologies

Through UNIX, a mechanism was introduced that allowed applications to communicate using a bi-directional byte stream. Applications that support this mechanism do so through a software abstraction known as *sockets*. Such applications communicate using a well defined protocol. A standard protocol is normally documented as an IETF RFC[1].

The Remote Procedure Call (RPC) [27] is a mechanism that allows a programmer to use standard procedure call syntax, but the execution of this procedure may take place on a remote machine. While RPC represents a higher level abstraction, like sockets, an intrinsic assumption is that communication involves only two parties.

The Object Management Group is a consortium that is committed to promoting an object-oriented approach to software engineering and establishing a reference model for distributed processing using an object-oriented methodology. This work culminated in the Common Object Request Broker Architecture (CORBA) specification [149]. Commercial implementations of CORBA have received some attention and are beginning to sell, albeit in small numbers. Although more advanced than the previous two technologies, CORBA is client-server based and hence is not equipped to support group communication [117].

The Parallel Virtual Machine (PVM) [73] and Message Passing Interface (MPI) [59] are technologies specifically aimed at the parallel programming community. These tools facilitate the development of parallel, typically numerical, applications that run on tightly coupled, massively parallel processing (MPP) machines. While support for process groups is provided the topologies available are oriented to map well onto mathematical representations, for example cartesian grids.

Reliable distributed computing has been the major goal for the ISIS system [26]. ISIS is a sophisticated toolkit that allows programmers to develop distributed applications using process groups. As the system makes extensive use of complex protocols to ensure fault tolerance, access to ISIS services is via API library calls. A success of open hypermedia systems, and in particular Microcosm, has been the ability to integrate the link service with desktop applications. Pearl [154] and Davis *et al* [52] argue that where the toolkit approach has limitations (see section 2.5.2), a more flexible model of integration can be achieved by providing access through a simple and open protocol. The Open Hypermedia Protocol (OHP) initiative [51] is seeking to define an open protocol to facilitate the integration of third party applications.

[1] Internet Engineering Task Force, Request For Comments

Comprehensive support for group communications can now be found within modern distributed operating systems, such as Amoeba [172]. Unfortunately, for the developer of an open hypermedia system to take advantage of these facilities presupposes that the end users of the hypermedia system will abandon their current, and perhaps favoured, operating system - which is highly unlikely. As Brown [29] observes, the hypermedia system developers must fit the world as it is rather than the world as we would like it to be.

6.6 The Design of the Heterogeneous Communication Model (HCM)

The remainder of the chapter is devoted to describing my design of a new Heterogeneous Communication Model (HCM). The design of the framework has sought to embody the salient principles of the DCM, but refines and extends them where necessary in order that it may function effectively within a heterogeneous, distributed, multi-user environment.

6.6.1 Process Addressing Scheme

For direct communication to take place one process must be able to address uniquely the process to which it wishes to send a message. This rudimentary requirement gave rise to a novel process addressing scheme. The addressing scheme, used within the HCM, provides a simple but extensible technique for use when processes register to receive messages and send messages.

As mentioned earlier, for the DCM Wilkins *et al* provide four methods for achieving communication between participating processes. It was identified that these four methods could be unified within a single addressing scheme using a metaphor borrowed from computer file systems. Each file in a file system can be uniquely addressed by using the fully qualified pathname, e.g., `/papers/hypertext/ht97.ps`. Similarly, a directory listing may be obtained that employs the use of *wildcards*. Both of these techniques were combined to produce a flexible process registration and addressing mechanism.

The process addressing scheme can support a *federated namespace* [156], allowing the naming context to be constructed from several distinct and independent naming contexts, as described below.

6.6.2 Customising the Addressing Scheme

The process addressing scheme forms the foundation upon which the support for message passing between processes executing on different machines is based. The first major benefit of the addressing scheme is that it can be customised by the architects of the host system to reflect the context in which it operates. This means that a process can send a message identifying the intended recipients using terms that have meaning within that context.

By way of a case study, the central components of Microcosm were analysed to identify how the addressing scheme could be mapped to a real system. The variables that differentiate typical Microcosm processes from one another were isolated in order to establish the fields. Primitive criteria by which a process within Microcosm could be identified are presented below:

1. the name of the process
2. the unique identifier of the process
3. the object being processed
4. the service provided by the process

Once agreed, all Microcosm process registrations would be required to conform to the following custom addressing template:

```
/ProcName/UniqueProcId/Document/Service
```

The first field is the name by which each Microcosm process can be referred to, e.g., *linkbase*. The operating system usually provides methods by which a unique identifier can be generated, e.g., the concatenation of a machine address and a process id. This number equates to the second field. The majority of Microcosm processes either render a multimedia document or process a file. The filename or the unique document identifier, assigned by the DMS, could represent the third field. All Microcosm processes provide a service, usually expressed through the *action(s)* they support. The action supported by the process could form the last field.

Similarly, the architects of other distributed systems would decide upon a set of identifying criteria that model the environment in which it is to operate. As the addressing scheme is customisable, the number of fields comprising the address and the semantics associated with each field can be dictated by the system designers.

6.6.3 Process Registration

As indicated previously, the addressing scheme is to be used when processes register both to receive messages and send messages. For a process to register with the HCM it must supply a registration entry denoting the type of message(s) that it is interested in receiving. Using the template for Microcosm defined above, two linkbase[2] processes could each register two service entries in the following style:

```
/linkbase/64.128.12675/lion.raw/follow.link
/linkbase/64.128.12675/lion.raw/create.link

/linkbase/65.156.8674/tiger.raw/follow.link
/linkbase/65.156.8674/tiger.raw/create.link
```

Once a process has submitted a valid registration entry to the HCM, it has formally declared an interest in receiving messages that may match on any of the criteria supplied in the entry. This remains the case until a process unregisters the entry.

6.6.4 Sending a Message

For a process to send a message it must supply both the message and the address of the intended recipient processes. The addressing mechanism provides scope for being as general or specific as is necessary when stating the destination address of a message. If every field of the destination process address is completed then direct process-to-process communication takes place. By placing a wildcard in one or more positions within the destination process address, a message can be accurately targeted at a specific community of processes. A Microcosm viewer process could direct a message to all linkbase processes that service *follow link* requests by quoting the following destination process address, where an asterisk represents a wildcard:

[2] A Microcosm linkbase stores links in files with an extension of `.raw`

```
/linkbase/*/*/follow.link
```

This addressing scheme allows conversations between processes to be conducted on a variety of levels. As can be appreciated, messages can be exchanged between Microcosm processes on a process name basis, a document basis or an action basis, or more usefully, combinations of all three fields. Although not quite as flexible as the HCM, a product called OrbixTalk [115] adopts an approach similar to the process addressing scheme for allowing *talkers* and *listeners* to communicate on a specific *topic*.

6.6.5 Location Transparency

The customisable process addressing mechanism mentioned above affords a high degree of process location transparency. Using the HCM as a communication layer, the initiating process can then identify and target the services of a remote process in a meaningful context, as opposed to first determining and then supplying the network addresses of remote processes.

6.7 Architecture Of The HCM

One of the key principles of distributed computing is that any processing need not be performed locally, but on the most appropriate machine. The design of the HCM was to embody this principle from the outset, but to support this, and the flexible message passing across machine boundaries described above, requires a slightly more complex architecture. While the logical model of the HCM can be seen in figure 6.4, one possible physical realisation can be appreciated in figure 6.5.

A *message router* process acts as a bureau at which processes, using the process addressing scheme as the vehicle, can dynamically advertise and withdraw their services. The concept of a user *session* was introduced to define a logical container for an array of processes registered with a message router. A message router is also responsible for coordinating all communication between participating processes wishing to send messages to other registered service providers. If necessary, per-process message queues are instantiated and maintained by a message router to ensure the integrity of message ordering in the system. A process may also query the current registration entries of a message router by supplying a pattern to be matched against. A pattern,

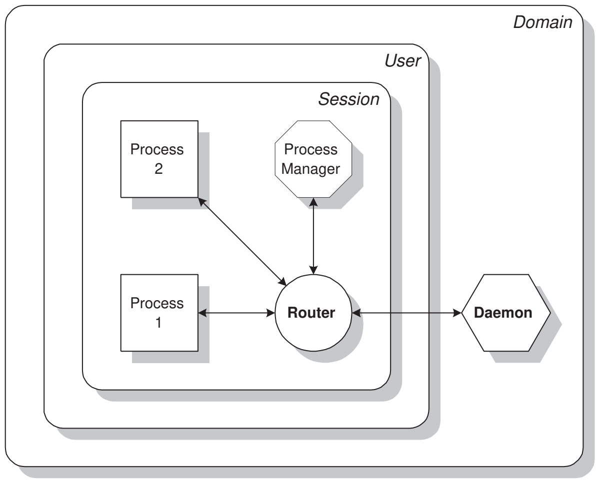


Figure 6.4: The logical model of a single user session.

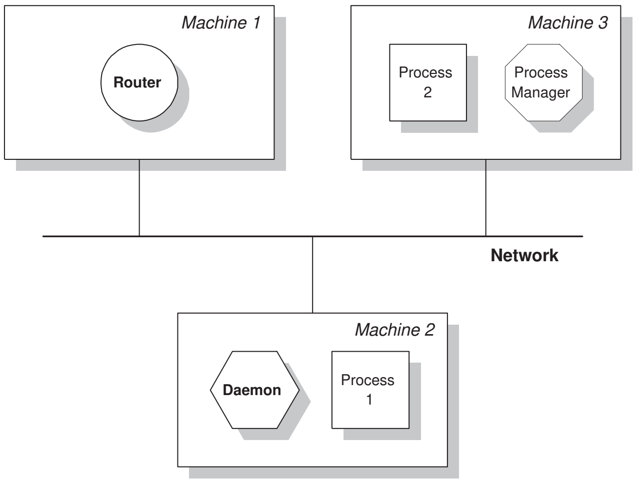


Figure 6.5: One possible physical realisation of a single user session.

---

in the shape of the processing addressing scheme, is sent to a message router and the entries matched are returned.

A *process manager* process also accompanies a message router as part of each user session. As might be expected, this eponymous process provides a degree of distributed process management. Distributed process invocation is achieved through the use of the remote shell (`rsh`), a standard UNIX utility that allows a program to be executed on a remote machine. Each machine able to service a `rsh` command has a symmetrical `rshd` daemon to receive the request. A version of the `rshd` was developed for Microsoft Windows to enable the process manager to perform distributed invocations on a non-UNIX platform in a transparent manner. If not overridden, the process manager uses a load balancing module to identify the least loaded machine upon which to spawn new processes. The load balancing module is provided via a shared library, allowing it to be replaced with an alternative algorithm.

Rather than each message router being restricted to pre-defined port numbers for communication, a single *domain daemon* process was introduced to serve each HCM *domain*. The notion of an HCM *domain* was introduced to denote a logical set of machines upon which a process manager can spawn processes. A domain daemon provides the sole fixed point of contact within each domain and every message router is required to register its network address with the domain daemon. The benefit of the domain daemon is twofold: message routers can dynamically allocate their network connections; the daemon can be queried to determine the network addresses of other message routers.

As mentioned in section 6.2.3, one of the essential requirements of this layer is that it should be open, and to remain as open as possible the HCM requires only a reliable transport mechanism. A network abstraction layer was jointly developed [46] to provide higher level support for socket-based communication. Theoretically, the network abstraction layer could reduce the effort of porting the layers to another network stack, but the current implementation has only been mapped onto TCP/IP. The layered construction of the HCM can be seen in figure 6.6.

The Microcosm message, described in section 4.3.3, has been proven to be extremely flexible as the unit of communication between processes. As such the Microcosm message was retained as the structure used by processes to communicate information through the HCM.

6.8 Extending The HCM to Support Inter-Session Communication

Once a process has lodged a registration with a message router that entry, or service, is then available to every process within the same session. A logical progression of

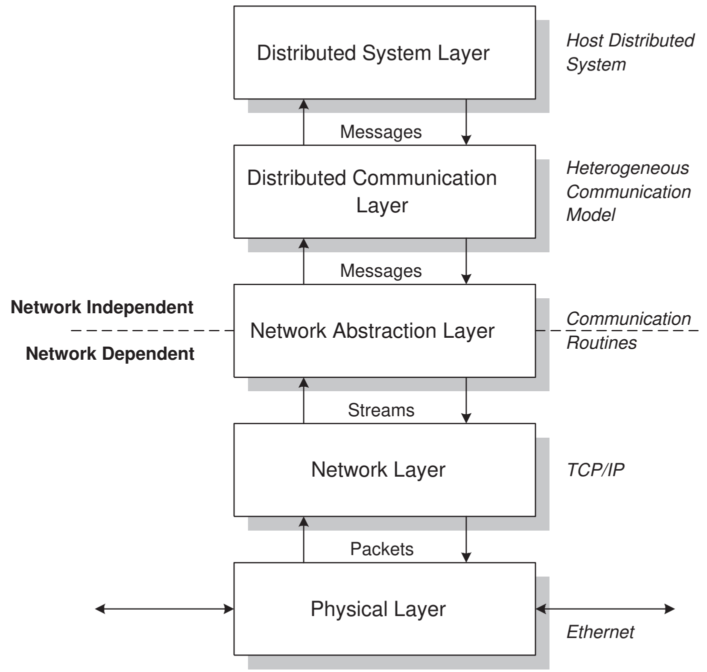


Figure 6.6: The layered construction of the HCM.

---

this is to allow remote processes mediated access to these services, and vice versa. A more thorough discussion can be found in [76].

6.8.1 Extending the Custom Addressing Template

In section 6.6.2, it has been described how the services provided by Microcosm viewers and filters could be mapped onto the process addressing scheme. The custom template proposed can provide a flexible method for uniquely addressing processes within the same user session. The following subsections describe how the custom template can be extended to support process group and inter-session communication.

6.8.1.1 Addressing Groups

A particular system may have a collection of processes that interact closely to perform a given task. An identity needs to be ascribed to such a community of processes allowing them to be addressed as a group. To facilitate this the template can be redefined as:

```
/GroupId/ProcessName/UniqueProcessId/Document/Service
```

The addition of this field provides a convenient mechanism by which a message could be targeted to an entire group of processes. A trivial, but useful, example could be to request all processes in a specified group to terminate.

6.8.1.2 Addressing Sessions

To enable processes in one user session to make requests of services on offer in another user session, the custom addressing template above needs extending further. As with the process group identifier, this is yet another trivial extension to the custom addressing template. This extra field will contain a unique identifier that denotes the presence of a session. The extended custom addressing template can be redefined as:

`/SessionId/GroupId/ProcessName/UniqueProcessId/Document/Service`

This provides the foundation for enabling HCM processes to address other HCM processes uniquely within multiple user sessions which could be physically located anywhere across the Internet.

6.8.2 Message Router Negotiation

Before inter-session communication can take place, a virtual connection must be established between the message routers within each session. This is followed by some preliminary negotiation. In section 7.7.2.1 a utility is described that allows a user to determine and select one or more of the services published by a remote session. Once selected, an exchange of process registration entries relating to the service(s) chosen takes place between the two message routers. When the negotiation has completed, the HCM processes are able to communicate across session boundaries in a seamless manner.

6.8.3 Providing Scalable Process Registration

A message router determines the recipient processes of a message by matching the destination process address against all of the process registration entries held internally. A simplistic solution to inter-session communication would be for a local message router to forward all messages to a remote message router in case any process in the remote session was registered to receive it. This approach is both inefficient and non-scalable, but if the local message router possessed some prior knowledge of the interests of the processes registered with the remote message router, it would be possible to determine whether or not the message should be forwarded.

A dichotomy exists between message routers supplying remote registration information to increase message passing efficiency, and message routers not supplying remote registration information but incurring unnecessary message transmission overheads. The following section presents contrasting approaches to remote registration.

6.8.3.1 Exhaustive Remote Registration

To ensure that only intended messages are transmitted in both directions between the message routers, it is necessary for each message router to have a copy of the other message router's appropriate, *published* process registration entries.

One distributed system may maintain a regular number of processes to satisfy the workload, while another may invoke and terminate processes on a frequent basis. The issue of remote registration arises when another process is started subsequent to the initial exchange of registration information between the message routers. It is likely that the registration information relating to this new process should also be propagated to the remote message router to denote its presence. The prototype was enhanced to fulfil this requirement to echo all new and published registration entries to the remote message router.

Although the goal of intelligent message routing was attained, the prototype affirmed that the overhead of communicating the registration information associated with a transient process would prove to be a prohibitive and non-scalable solution. This problem would be further exacerbated when considering that the user may also have forged virtual connections to a number of remote message routers, each one requiring the registration information of the new process.

6.8.3.2 Imprecise Remote Registration

Tanenbaum [172] argues that effective message routing can still be achieved with incomplete information. The computer file system metaphor upon which the process addressing scheme is based allows a process to express a degree of imprecision by specifying a *wildcard* in the appropriate position(s). It was observed that this feature was also the most natural and obvious mechanism for representing process registration entries within remote message routers. Consider the following example featuring several familiar Microcosm processes. The three processes exist within the same session and belong to the same logical group *classical*. The two *linkbase* processes operate upon the *mozart* and *bach* linkbases, while the *complink* process operates upon the *composer* index file. The first process offers three services while the other processes both offer two.

```
/64.128.6397/classical/linkbase/64.16.5248/mozart.raw/edit.link
/64.128.6397/classical/linkbase/64.16.5248/mozart.raw/follow.link
/64.128.6397/classical/linkbase/64.16.5248/mozart.raw/create.link

/64.128.6397/classical/linkbase/64.16.7264/bach.raw/find.buttons
/64.128.6397/classical/linkbase/64.16.7264/bach.raw/follow.link

/64.128.6397/classical/complink/64.96.8539/composer.idx/find.phrase
/64.128.6397/classical/complink/64.96.8539/composer.idx/index.docs
```

The prototype was subsequently extended to enable a message router to factor out the common elements from registration entries. The above registration entries could be encompassed by the following single entry:

```
/64.128.6397/classical/*/*/*/*
```

Where previously a local message router would have had to supply a remote message router with the six registration entries above, a single wildcarded entry will now suffice. The problem of the frequent need to propagate the remote registrations of transient processes can be resolved by applying this principle of genericity. For example, if a further *linkbase* process was started advertising the following registration entries:

```
/64.128.6397/classical/linkbase/64.24.9834/brahms.raw/find.buttons
/64.128.6397/classical/linkbase/64.24.9834/brahms.raw/follow.link
```

It is clear that this process has a similar set of registrations to those declared previously, and as such will not have to be propagated to a remote message router(s) as they are already subsumed within the generalised entry factorised earlier by the message router.

6.8.3.3 Compensating for Genericity

It has been recognised though, that the process of factoring can produce an over generalised remote registration entry which may result in a message router receiving messages but with no local processes registered to receive them. This eventuality can be circumvented through the use of heuristics to monitor the number of unwanted messages and for appropriate measures to be taken should this figure exceed an unacceptable threshold. Consider a situation where the user starts an additional process subsequent to the exchange of registration information between the message routers. This additional linkbase declares the following registration entry:

```
/64.128.6397/jazz/linkbase/64.8.2461/parker.raw/create.link
/64.128.6397/jazz/linkbase/64.8.2461/parker.raw/follow.link
```

It is possible that a message router could re-issue its remote registration to reflect this change by increasing the generality. This could be accomplished with the following:

```
/64.128.6397/*/*/*/*/*
```

As suggested earlier, if this remote registration resulted in an excess of unwanted messages being received by the message router, it would be prudent to express the remote registration as two marginally less general entries. This could be achieved using the following entries:

```
/64.128.6397/classical/*/*/*/*
/64.128.6397/jazz/*/*/*/*
```

An adaptive remote registration mechanism can be provided by incorporating simple heuristics which monitor both the flow of messages and registration entries over time and aim to achieve an efficient compromise. The first simple heuristic, 6.10, can be used to decide whether the precision of the remote registration entry needs to be increased. This is based upon whether the proportion of erroneous remote messages received exceeds a given threshold over a specified window of time.

$$\frac{E_{(t+\alpha)} - E_t}{M_{(t+\alpha)} - M_t} > T \tag{6.10}$$

Where E is the number of erroneous messages received, M is the total number of messages received, t is the current time, α is the size of the window and T is an arbitrary threshold.

The second simple heuristic, 6.11 provides the counter case. It is used to evaluate if the precision of the remote registration entry needs to be decreased. The ratio of re-issued remote registrations to the remote messages received is compared with a given threshold over a specified window of time.

$$\frac{R_{(t+\alpha)} - R_t}{Q_{(t+\alpha)} - Q_t} > T \tag{6.11}$$

Where R is the number of remote registrations received, Q is the total number of remote messages received, t is the current time, α is the size of the window and T is an arbitrary threshold.

The strength and versatility of the process addressing scheme has been illustrated by showing how it can provide a unified and scalable mechanism for both local and remote process registration.

6.9 Alternative HCM Configurations

The extensions reported above enhance the flexibility of the HCM and allow it to be used in a variety of configurations:

Single user environment seen earlier in figure 6.4.

Multi-user environment whereby processes within one user session may communicate with the processes of another user session but within the confines of a single domain.

Widely distributed multi-user environment whereby processes within one user session may communicate with the processes of another user session, across disparate domain boundaries. This can be seen in figure 6.7.

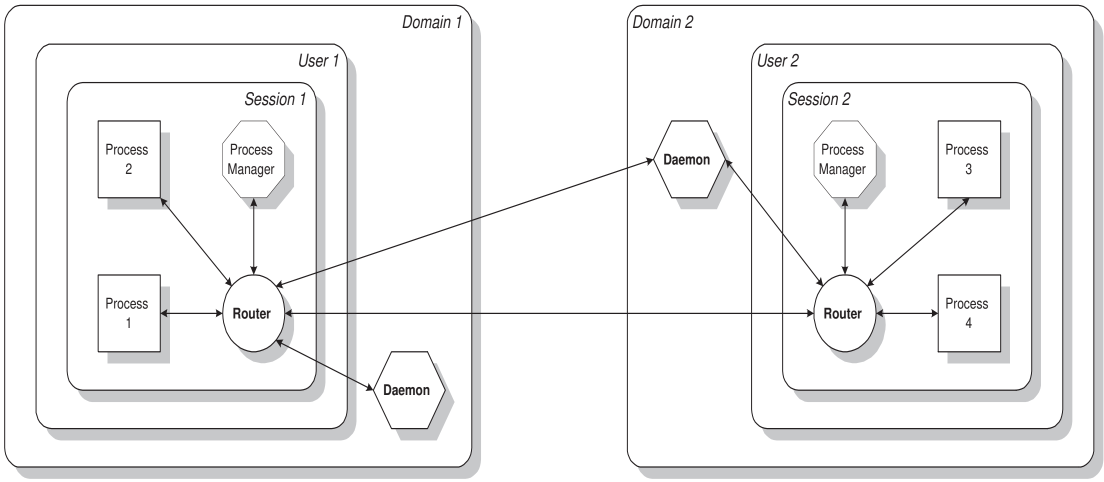


Figure 6.7: The HCM providing a widely distributed multi-user environment.

6.10 Status of the HCM

In addition to Microcosm TNG, the HCM is used with the MultiDOSm system [191], a prototype framework for an open Graphical Information System [162] and successfully trialed with MAVIS [131]. The design of the HCM message router has influenced the design of the MEMOIR architecture [113], a European project for enhancing access to corporate knowledge.

The HCM framework has so far been ported to PCs running Linux, Silicon Graphics running IRIX, IBM running AIX, Sun running SunOS and Solaris. Although the framework can be easily accessed via the open protocol, a library has been provided with language bindings for both C and Java.

6.11 Summary

Three alternative models of communication have been proposed for use within Microcosm. Each model was examined to determine the extent of its scalability and flexibility and it was concluded that the DCM offered the most promise. The DCM was then deconstructed to identify suitable approaches for applying these principles to a distributed environment.

The design of the HCM has embraced many of the requirements of a distributed communication layer and provides an open framework for the development of distributed systems. It provides a context sensitive process addressing scheme above a scalable message passing solution. In addition, the heterogeneous nature of the framework promotes the use of available resources, allowing processes to be allocated to machines with the most suitable facilities, or which are being under utilised.

A novel process addressing scheme has been described, providing a symmetrical mechanism for specifying both process registration and message destinations. Firstly, it was shown how the generic process addressing scheme can be tailored to model the domain in which the target system is to operate. This affords the system architects the total freedom to assign simple or complex semantics to a custom template. This means that the granularity at which processes may interact is completely configurable. The added flexibility provided by the wildcard feature allows a process to define the scope of the audience over which a message is to be directed. This facilitates conversations to be conducted on a variety of levels, dependent upon the richness of information incorporated within the addressing template. Secondly, it was illustrated how the same process addressing scheme was used to provide an adaptive and scalable technique for remote registrations. The strength and versatility of the process addressing scheme has been vindicated by showing how it can provide a unified mechanism for both local and remote process registration. Thus it has been demonstrated that the HCM provides a scalable and seamless architecture for supporting process interaction across a local network or the Internet.

The client/server model, the predominant architecture among systems using the Internet, can restrict the development of responsive distributed systems. In its basic form, the client/server paradigm makes it difficult for a server to take the initiative in delivering new events to the client. Furthermore, some processes need to act as both client and server depending upon whom they are interacting with. The HCM allows applications to benefit from a flexible peer-to-peer alternative.

The next chapter describes the design and architecture of a distributed open hypermedia system, called Microcosm TNG, that has been built above the HCM described in this chapter.

7 Microcosm TNG: A Model for Distributed Open Hypermedia

7.1 Introduction

In chapter 5 a number of hypermedia systems were presented and contrasted along the dimensions of openness, flexibility and distribution. These systems can be divided into two categories: those that rely upon the infrastructure of the WWW and those that do not. While it is necessary to investigate the potential and limitations of the WWW infrastructure, it is also valid to explore alternative architectures and to develop new and open solutions for distributed information management.

In this chapter the requirements of an open hypermedia system introduced in section 2.8.1 are revisited, but explored with the additional context of distribution. These requirements provided the motivation for the development of a distributed, open hypermedia system called Microcosm: The Next Generation (TNG). The remainder of the chapter is devoted to describing its design.

7.2 Requirements for Distributed Open Hypermedia

A six point working definition of open hypermedia by Davis was reviewed in section 2.8.1. Each point is revisited in this section to examine the specific effect of distribution, if any, upon these requirements.

An open hypermedia system is a hypermedia system where the term open implies the possibility of importing new objects into the system. A truly open hypermedia system should be open with regard to:

1. *Size:*

   It should be possible to import new nodes, links, anchors and other hypermedia objects without any limitation to the size of the objects or to the maximum number of such *objects that the system may contain, being imposed by the hypermedia system.*

For open hypermedia systems of the future to support application domains such as digital libraries and industry, it has been noted that the design of such systems must be scalable [147]. Scalability in this sense relates to the ability of the system to manage effectively large numbers of users, tools and stored objects [183]. An issue intrinsic to the management of large numbers of distributed objects is that of naming. For each object to be referenced by the system it must be uniquely addressable, and schemes for addressing large distributed information repositories must also be flexible and scalable.

Malcolm *et al* reminds us that information in these environments is likely to be dynamic, regularly being revised and modified. Notification of new information and changes to transient information must be propagated throughout the system and brought to the attention of interested parties. This is especially important in large distributed information spaces, as no user can hope to be intimately acquainted with massive volumes of data.

2. *Data formats:*

   The system should allow the import and use of any data format, including temporal media.

   It is increasingly common for personal computers and workstations to be equipped with multimedia hardware able to play temporal, or continuous, media such as digital audio and video. Although this technology is suitable for playing continuous media stored locally, delivering continuous media over the Internet is a non-trivial exercise. While projects such as Vosaic [41] and AudioFile [128] demonstrate that it is technically possible, techniques for improving quality remain a research issue as the current infrastructure of the Internet is less than ideal.

   Various researchers are confronting the issues of specifying and delivering synchronised hypermedia documents [104, 32]. Of the many proposed solutions to this non-trivial task, the most favourable is to provide support amongst the layers of the operating system. Existing mechanisms have proved inadequate on their own so techniques such as triggering have been introduced to complement existing services. These problems are further exacerbated when considering the mechanisms required for synchronous coordination within a heterogeneous and distributed environment.

3. *Applications:*

   The system should allow any applications to access the link service in order to participate in the hypermedia functionality.

   Enabling the original editor, used to create the document, to be used also by the link service to present the document remains as one of the goals of open hypermedia. The notion of cross-application linking is practically synonymous with open hypermedia. Successes in this area has been demonstrated with a number the open hypermedia systems, described in chapter 3. Many of these systems allow links to be created and followed between third party applications running on the same genre of operating system. It is becoming increasingly common though for information workers to interact with a variety of applications hosted on a diverse range of computing platforms. Trigg *et al* [174] describe a scenario requiring Søren to create a link between a Microsoft Windows text document and a UNIX Oracle database. As such, it is necessary for contemporary distributed open hypermedia systems to support cross-platform linking through heterogeneous application integration.

   Although this point is largely concerned with integrating desktop tools, it is clear that users also access the information and services available on remote systems. The ability to integrate with existing Internet information services such as the WWW, Gopher and FTP and utilise these in conjunction with their favoured hypermedia system is crucial.

4. *Data models:*

   The hypermedia system should not impose a single view of what constitutes a hypermedia data model, but should be configurable and extensible so that new hypermedia data models may be incorporated. It should thus be able to interoperate with external hypermedia systems, and to exchange data with external systems.

The issue of hypermedia interchange, discussed in section 2.7, is complex and the subtleties of the various hypermedia data models continue to make the task of translation non-trivial. While a degree of integration has been demonstrated between some open hypermedia systems - Microcosm with Guide [107], HyperDisco with Chimera [186] and several with the WWW [39, 160, 87, 5] - the level of interoperability is less than satisfactory.

The link service protocols used by open hypermedia systems are currently all proprietary, but the OHP initiative, discussed in section 2.8.3, aims to abstract the successful themes to arrive at a standard open hypermedia protocol. The establishment of standard protocols can only aid interoperability.

5. *Platforms:*

   It should be possible to implement the system on multiple and distributed platforms.

   In striving to achieve ever closer integration with the desktop environment, some open hypermedia system designers have chosen to exploit any potentially useful facilities that a particular platform has to offer. While this can obviously be advantageous for a specific implementation, such a philosophy destroys any opportunity for developing a cross-platform solution. An ideal open system should be capable of offering an identical range of services across a variety of platforms, but supporting this ideal requires portable source code. Such an ideal would preclude the use of platform specific techniques.

   As greater demands are placed upon hypermedia systems to support more computationally intensive link resolution and media-based querying, such as MAVIS [131] and Miyabi [112], it is becoming necessary to investigate distributed architectures as a way of providing more flexible, cost-effective and scalable solutions.

   Hill [109] argues that the current generation of distributed hypermedia systems are strongly tied to a client/server model and are insufficiently flexible to allow a range of configurations. He argues that a peer based model is likely to provide greater versatility.

6. *Users:*

The system must support multiple users, and allow each user to maintain their own private view of the objects in the system.

The ability to publish and access information is one of the central benefits of distributed information systems. However, the information generated by users tends to be clustered by physical location rather than by content, presupposing that users know where to look for desired information. This effect of *balkanisation* was observed by Nelson [144]. Noll *et al* [147] recognise the need for supporting resource location transparency in hypermedia systems, allowing a document to be uniformly accessed whether local or remote.

Reviewed in section 5.6.6, the extensions to Microcosm made by Hill [110] allow users to make hypermedia services within their session available to remote users, and vice versa. The notion of sharing is extended beyond that of just documents to incorporate both hypermedia information and services.

Following the review of contemporary distributed hypermedia systems in chapter 5, three primary requirements have been identified.

distributed information sharing Distributed access to documents is the most fundamental requirement and is supported by all the systems reviewed, albeit with varying degrees of location transparency.

distributed information organisation This is the ability to present logical collections of physically distributed documents in a unified fashion. Explicit support for this can be seen with Hyper-G collections [7].

extensible distributed services This allows the user to customise their hypermedia environment by incorporating an eclectic range of distributed services. This feature is exemplified by Microcosm [110].

The aim when designing Microcosm TNG was to build a framework for an open and extensible distributed hypermedia system which would embody these three core features.

7.3 Architectural Constraints Upon Microcosm

It is beyond doubt that the philosophy behind the Microcosm project was well-founded and that the architecture developed has significantly contributed to the field of open hypermedia. In this section a number of situations will be described for which the architecture proved insufficiently flexible.

When a Microcosm session is initiated, the user is required to select a single hypermedia application from a list. This application then defines the context for the information that the user will access. Although the user may embellish the session with personal resources, the fundamental limitation lies in the inability of the architecture to accommodate multiple active hypermedia applications within a session.

As discussed in section 6.3.1.4, the nature of the filter chain presupposes that each message passes from filter to filter in sequence. Filters have been developed that must reside at a specific position in the chain in order to function correctly. The *linker* (link creation filter) must precede all *linkbases* (link database filters), while the *available links* (link destinations filter) must succeed all *linkbases* in the filter chain. Users must be aware of the implications of the ordered filter chain, the ramifications of which are described below.

Microcosm, like many contemporary open hypermedia systems, allows multiple linkbases (webs, or contexts) to be active within a user session. Other than the need for additional control messages as explained in section 6.3.1.2, the serial nature of the filter chain does not present a problem when a *follow link* request is issued. The request is propagated to each linkbase in the chain giving it an opportunity to respond. The same is not true though during the link creation process. After selections have been made, one or more start and end anchor messages are sent by the viewers and are intercepted by the linker filter positioned near the head of the chain. The linker filter then displays a dialog containing the details of the link to be created. If the user accepts these details the linker packages them into a *create link* request and sends it to the next filter in the chain. To prevent the link from being created in every linkbase in the chain, the linkbase positioned earliest in the chain assumes the responsibility for creating the link and subsequently destroys the message. The consequences of this limitation have meant that:

- when creating links within a large-scale hypermedia application comprising multiple linkbases, it is often necessary to rearrange the positions of linkbases within the filter chain in order to create a link in the desired linkbase.

- the problem described above could be circumvented by employing only a single linkbase. This solution is only suitable for small applications, as a non-modular approach can hinder the task of link maintenance.

- in a didactic context, the first linkbase is considered to be the *student's linkbase*, in which personal links and annotations are collected, while the linkbases in subsequent positions contain links created by the lecturer.

The system does not allow the user to specify simply the linkbase in which new links are to be stored, and, as a consequence, prevents the user from having multiple, personal linkbases active. The examples above illustrate how the serial nature of the filter chain has required some inelegant compromises to be sought.

Although the designers of Intermedia recognised that having only a single web proved too restrictive, an idea suggested, but not attempted, by Haan *et al* [96] was for a link that could open a new Intermedia web. Again, although not yet attempted with Microcosm, the architecture would not preclude this. What Microcosm does preclude is the capability of a link to open a new hypermedia application, thus augmenting the session with another related context. This limitation is because Microcosm cannot support multiple, active hypermedia applications.

Introduced in section 4.9, the Document Management System (DMS), or *docuverse*, maintains a single database of all the documents known to Microcosm. Ideally, the DMS would be situated in the filter chain and accessed in an identical manner to the rest of the hypermedia services on offer, but, as explained in section 6.3.1.4, the serial nature of the filter chain prohibits a filter from holding a bi-directional conversation with another filter. To overcome this limitation the DMS is implemented as a shared library and the components of the system access the services through an API. This design decision has a number of shortcomings and will be explored further, along with related issues, in section 7.9.

Serious doubts as to the scalability of the filter chain were raised in section 6.3.1.2. In addition, the components of a Microcosm session cannot be distributed and are restricted to executing on a single machine. While the approach taken by Hill [110] allows the sharing of hypermedia services, through the publication of filters, it does not provide a solution for distributing a single user session. Rather than allowing a user to run individual components of their Microcosm session on disparate machines, it constrains them to running all of these components on the local machine but with the opportunity of accessing additional services offered by remote Microcosm sessions. The scalability of Hill's distributed version of Microcosm is compared with that of Microcosm TNG in section 7.7.3.

7.4 Redefining the Notion of a Hypermedia Application

The model adopted by the WWW is one in which all entities are considered to be documents; even CGI scripts exist to generate new documents. As the granularity of the entire system exists at the document level, it is trivial for the user to bind documents, through links, into shareable information resources. Equally though, as no intrinsic mechanism for abstraction is available, it is difficult to arrange individual resources into logical collections organised by context.

As mentioned in section 7.3, Microcosm has the notion of a *hypermedia application* which loosely defines the processes and link databases that comprise a single user session. I have reworked this weak specification of a hypermedia application to define a mechanism for encapsulating closely related resources. A hypermedia application is now defined to be *a binding of a logical grouping of documents, the services that operate on the collection and all associated configuration details*.

This definition enables the scope of a resource base to be limited to a defined and manageable entity. It also provides a convenient abstraction that allows each hypermedia application to be considered a self-contained, publishable and shareable entity. This enhanced model allows entities to be associated with a hypermedia application that possesses context, while also being able to retain the desirable properties that exist at the document level.

The relationship between users, sessions and hypermedia applications can be appreciated in figure 7.1. It can be seen that a user may have multiple sessions each containing multiple applications which can, in turn, be shared among multiple users.

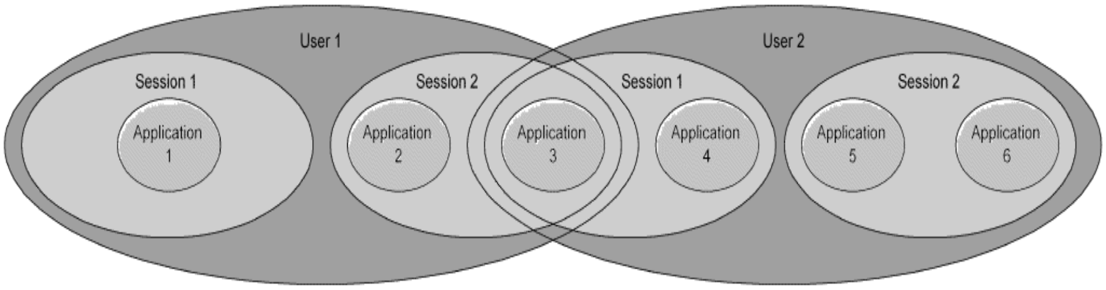


Figure 7.1: The relationship between users, sessions and hypermedia applications.

This enables published applications to be shared from and assimilated within user sessions to construct richer information spaces, rather than being restricted to a solitary application as before. As the author of a hypermedia application is now afforded a level of abstraction beneath which any configuration issues may be concealed, remote users of such an application need not be concerned with configuration, but can simply augment their user session with another logical entity.

The revised definition and conceptual relationships between users, sessions and hypermedia applications provides a suitable model with which to address the three key requirements of distributed information sharing, distributed information organisation and extensible distributed services, identified in section 7.2. Selected information resources and configurable services are bound to a hypermedia application, which, as a publishable entity, can be shared by multiple users. The following sections describe the architecture to support this model.

7.5 The Conceptual Model of Microcosm TNG

Both the Dexter Model and the Microcosm Model, introduced in sections 2.3 and 4.2 respectively, are composed of three separate layers. Many similarities between the Microcosm Model and the Dexter Model exist at each of the three layers. As Microcosm TNG can be considered to be an evolution of Microcosm, it remains valid to ascribe the abstractions of the three layered conceptual model to Microcosm TNG.

The models noted in the preceding paragraph identify the three primary facets of an open hypermedia system to be that of a presentation component, a link management component and a document management component. One particular avenue of this research has been to raise the profile of the Document Management System within the context of open hypermedia. In a distributed open hypermedia system it is necessary for the Document Management System to occupy a more important role than before. Advantages can also be gained by making the Document Management System a more interactive component of the open hypermedia system. As such, an alternative view of the relationship between the three primary facets can be appreciated from figure 7.2. This figure illustrates that each facet is of similar importance and connected to the other two, creating the impression of a peer model.

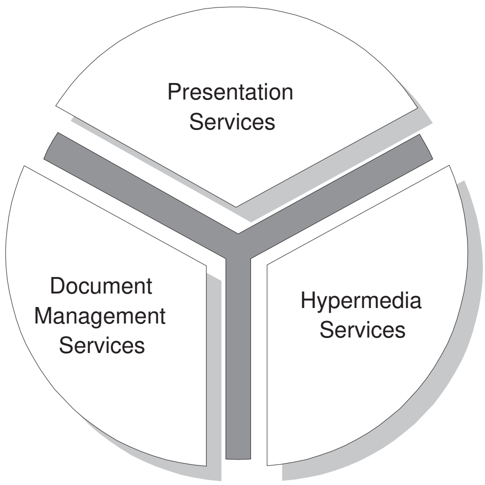


Figure 7.2: Combining the three principal facets of an Open Hypermedia System.

7.6 The Architecture of Microcosm TNG

As alluded to in section 7.5, Microcosm and Microcosm TNG share a number of the same ideals. It was considered essential that the core philosophies of Microcosm should be inherited by Microcosm TNG. The architecture of Microcosm TNG can be considered superficially similar to that of Microcosm, in that a presentation front end provides access to a multitude of back end services. Central to the design of the architecture are the three key requirements of distributed information sharing, distributed information organisation and extensible distributed services, introduced in section 7.2.

7.6.1 Modelling Microcosm Using Microcosm TNG

Although the architecture of Microcosm TNG [81, 80] represents a significant departure from that of Microcosm, it is an evolution of the original concepts and thus is able to model much of Microcosm. It is possible for a Microcosm TNG user to create a session containing only a single hypermedia application and to configure the system to constrain all processes to execute on the same machine.

7.6.2 Process-driven Infrastructure

Many hypermedia systems demand that the user start the core infrastructure, which may in turn invoke other components, prior to using the system. While this route into Microcosm TNG is available, an alternative approach permits a process to trigger the start of the system infrastructure autonomously. This feature allows the system to become more process driven. By empowering an HCM-aware process to activate the framework it allows the core components to appear more transparent to the user. As a consequence, the user requires less explicit interaction with the framework, allowing it to assume a less obtrusive role within the desktop environment.

7.6.3 Managing Dynamic Distributed Processes

The fundamental role of the *process manager* is described in section 6.7. Judicious distributed process invocation makes it possible to harness latent processing power, hence allowing better use of the available resources. The process manager can also be configured to assign certain tasks to certain machines. For example, link database processes and image processing programs can be designated to run on powerful servers, while presentation viewers may need to be constrained to executing on the local machine. If unconstrained, the process manager proactively distributes all processes associated with a user session. The Chimera [6] system has a *process invoker* that performs a similar role to the process manager, although it is unclear as to whether proactive distribution across heterogeneous machines can occur.

An optional, but valuable, process is the *message logger*, primarily developed to aid with debugging Microcosm TNG. It registers to receive *every* message, and upon receipt of a message records the content. The message logger essentially provides persistent storage for messages. Messages can be a valuable commodity as they provide useful information about the events that took place within the system. For example, a user may incorporate one or more of the navigation aids within their user session. These tools intercept certain types of messages and analyse their content in order to suggest any relevant links and documents. As new services can be started at any time, it is conceivable that a navigational device may be started midway through a session if the user experiences disorientation. Under Microcosm, a filter started at a later time is not proxy to messages generated before it had been started. With Microcosm TNG, processes started at any time during a session can requisition from the message logger any messages that were issued prior to their activation. By exploiting this service, processes can be rapidly imbued with *session context* through a history of appropriate messages.

7.6.4 Microcosm TNG Custom Addressing Template

The notion of a custom addressing template was introduced in chapter 6 with reference to the HCM. The Microcosm TNG model was mapped to a suitably rich custom addressing template. The following template is the one to which all the processes in the system adhere:

```
/SessionId/ApplicationName/ProcessName/ProcessId/Document/Service
```

This template translates to: the user session identifier to which this process belongs; the hypermedia application name to which this process belongs; the name of the Microcosm TNG process; the identifier of the Microcosm TNG process; the document upon which this process operates; a service which this process offers.

7.7 Distributed User Sessions

Support for process distribution in Microcosm TNG is provided by the HCM which, as described in section 6.9, is flexible enough to allow three different configurations. In addition to explaining the various configurations, a mechanism is described that enables remote applications to be browsed and incorporated into a user session.

7.7.1 A Distributed Single User Session

Even at the embryonic stage, it had always been intended that Microcosm would be extended in order to allow it to operate in a distributed environment. The modular viewers and filters are chiefly responsible for the extensible functionality offered by Microcosm. The self-contained nature exhibited by these processes predisposes them to being distributed over a heterogeneous network of machines.

The PROXHY system [119] is characterised by its object based process model. Although the fine grained PROXHY approach proved versatile and extensible, it incurred a serious performance penalty. By contrast, the granularity of distribution in the Microcosm TNG model is far coarser, occurring at the process level.

While the logical model of a typical Microcosm TNG single user session can be seen in figure 7.3, one possible physical realisation can be appreciated in figure 7.4.

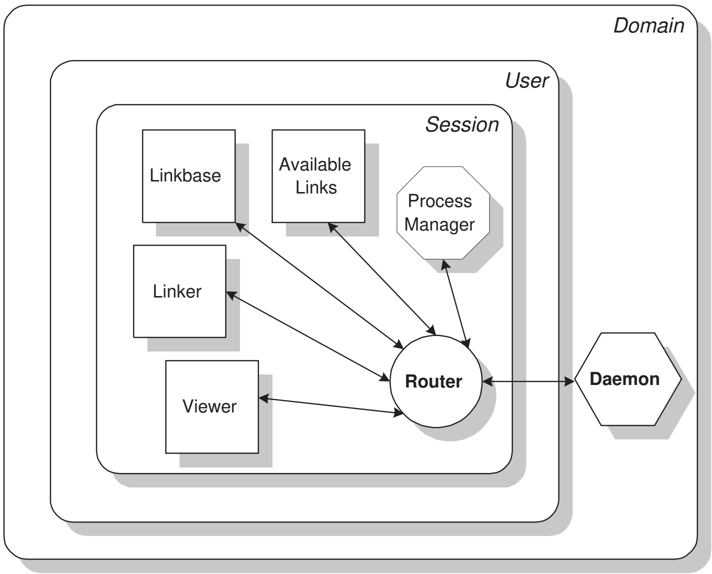


Figure 7.3: The logical model of a Microcosm TNG single user session.

7.7.2 Multi-user Distributed Sessions

The sharing of distributed hypermedia applications between Microcosm TNG users can either be restricted to those located in the same domain, or widened to permit connections from users in disparate domains [81, 80].

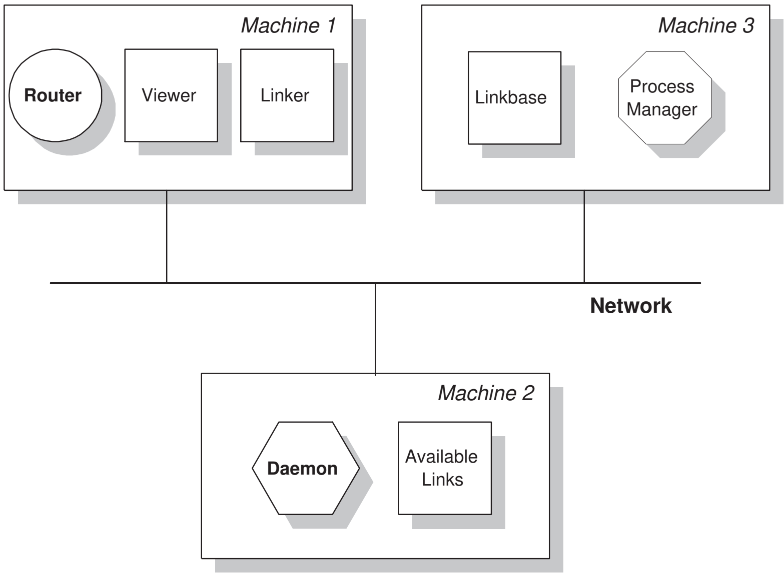


Figure 7.4: One possible physical realisation of a Microcosm TNG single user session.

7.7.2.1 Accessing Remote Hypermedia Applications

The WWW has no intrinsic notion of either a global map of all documents (hypergraph) or a location mechanism. As such, before a remote document can be retrieved and read the URL must first be determined. This can be obtained either by word-of-mouth, electronic mail, traversing a link or as a result of employing one of the many search engines. Similarly, before a user can make direct use of a remote hypermedia application they must determine the network address of the domain within which it is being offered. As hypermedia applications are referenced with respect to a domain they are less volatile than documents referenced by a URL, which also have an associated pathname.

A domain daemon provides a fixed point of contact within a domain and can also provide a fixed point of contact to users outside a domain. An optional utility, called the *app browser*, has been developed to facilitate the browsing of remote domain daemons to allow users to discover more information regarding any hypermedia applications available. The user interface to this utility can be seen in figure 7.5.

A user indicates an interest in browsing a domain daemon by pressing the locate button. The *app browser* then presents a *hot list* of domain daemon addresses from

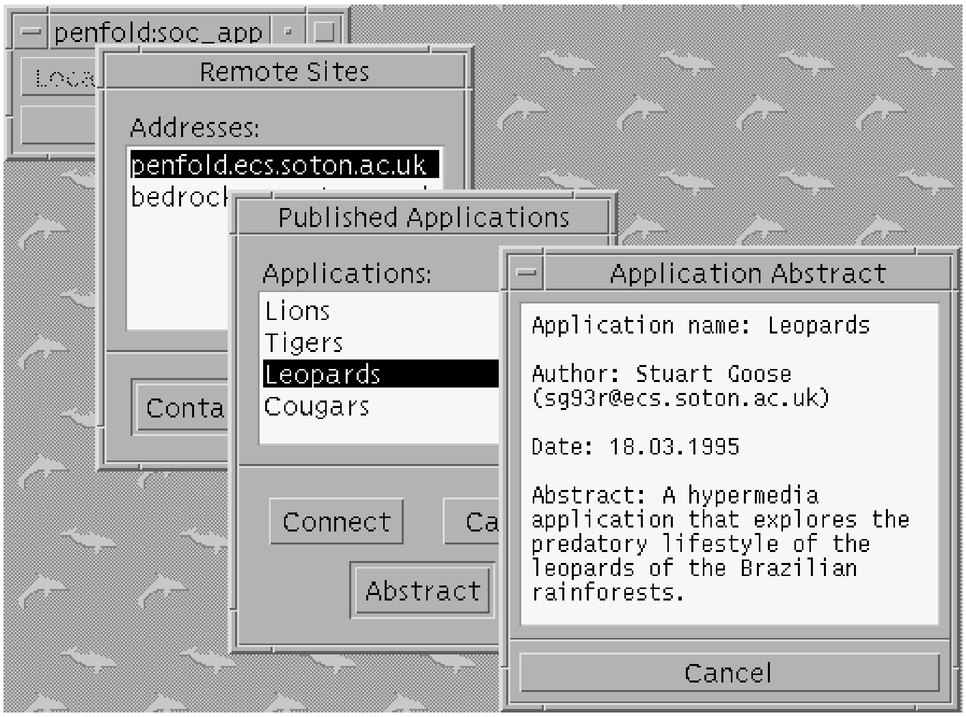


Figure 7.5: A screenshot of a utility for browsing and connecting to remote hypermedia applications.

---

which the user can select. The chosen domain daemon is then contacted and sent a message requesting a list of all available hypermedia applications. On receipt of this request, the domain daemon asks each message router registered with it to reply to the originating process listing all of the hypermedia applications published from them. The list of available hypermedia applications returned is presented to the user by the *app browser*. The user can obtain pertinent information and a synopsis of an application by pressing the abstract button. One or more of the hypermedia applications can then be selected from the list box and assimilated within the user session by pressing the connect button. When the connect button is pressed, the user's local message router exchanges process registration information with one or more remote message routers providing the hypermedia applications, as described in section 6.8.2. Except for network latency, the services of the remote hypermedia applications can be used in an identical manner to those of a local one. With reference to point 6 of the requirements in section 7.2, this approach illustrates a method for providing location transparent hypermedia services.

A logical model of the Microcosm TNG architecture is illustrated in figure 7.6. This shows that User A in Domain 1 has browsed Domain 2 and established a virtual connection to User B's session in order to access the remote hypermedia application about Lions.

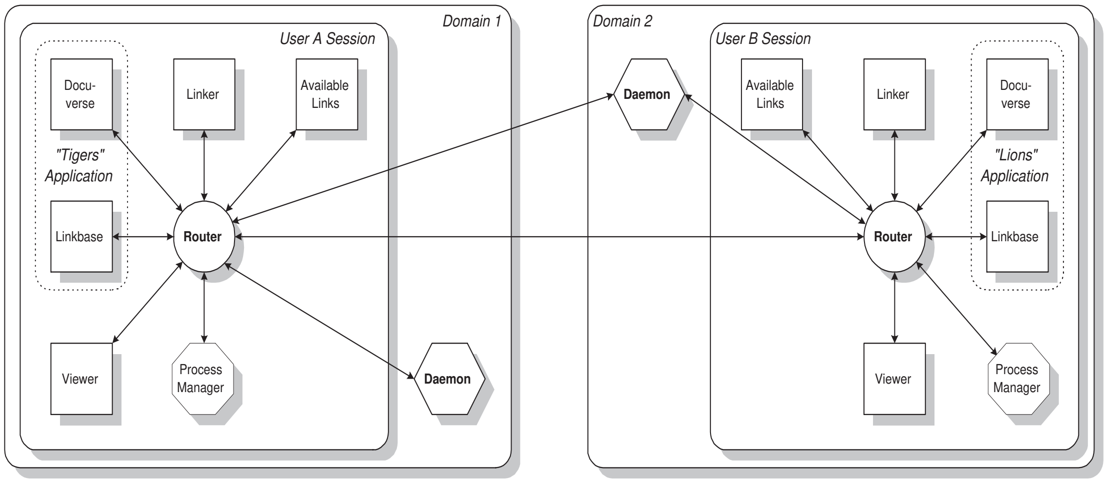


Figure 7.6: Microcosm TNG users sharing hypermedia applications in a distributed environment.

7.7.2.2 The Hypermedia Application Abstraction

Although Hill [109, 110] describes how individual Microcosm filters can be published, when untrained users are presented with a list of filters from which to choose it is likely that they will experience problems. The most obvious problem is that an individual filter possesses little context, which may leave the user unsure as to the role and the relevance of the service. This model also demands an understanding of the implications of the filter chain, requiring all users to be sufficiently skilled at reconfiguring the system to accommodate additional remote filters.

By introducing the higher level abstraction of a hypermedia application and employing a position independent distributed communication model, the problems mentioned above are eradicated. While browsing, the user can deduce the context of a hypermedia application from the name and the synopsis provided by the authors. A simple connect operation from the *app browser* gives access to an arbitrary number of hypermedia application services, the specifics of which are concealed from the user.

7.7.3 Comparison with Extended Microcosm

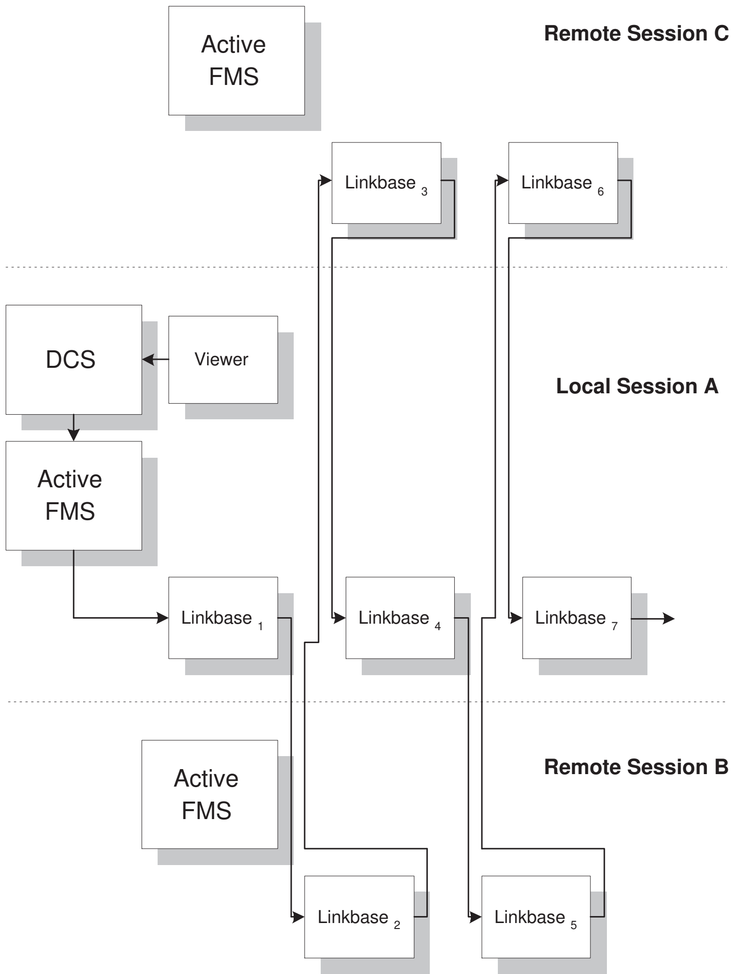


Figure 7.7: Extending Microcosm to facilitate a distributed filter chain.

Although the communication models for Microcosm were analysed in chapter 6, the comparison did not examine the distributed operation as not all of the models addressed this requirement. As such, the two distributed architectures proposed for Microcosm are briefly contrasted in this section through an example. A user has a **Local Session A** which has three linkbases and in addition is connected to **Remote Session B** and **Remote Session C**, each providing access to two further published linkbases.

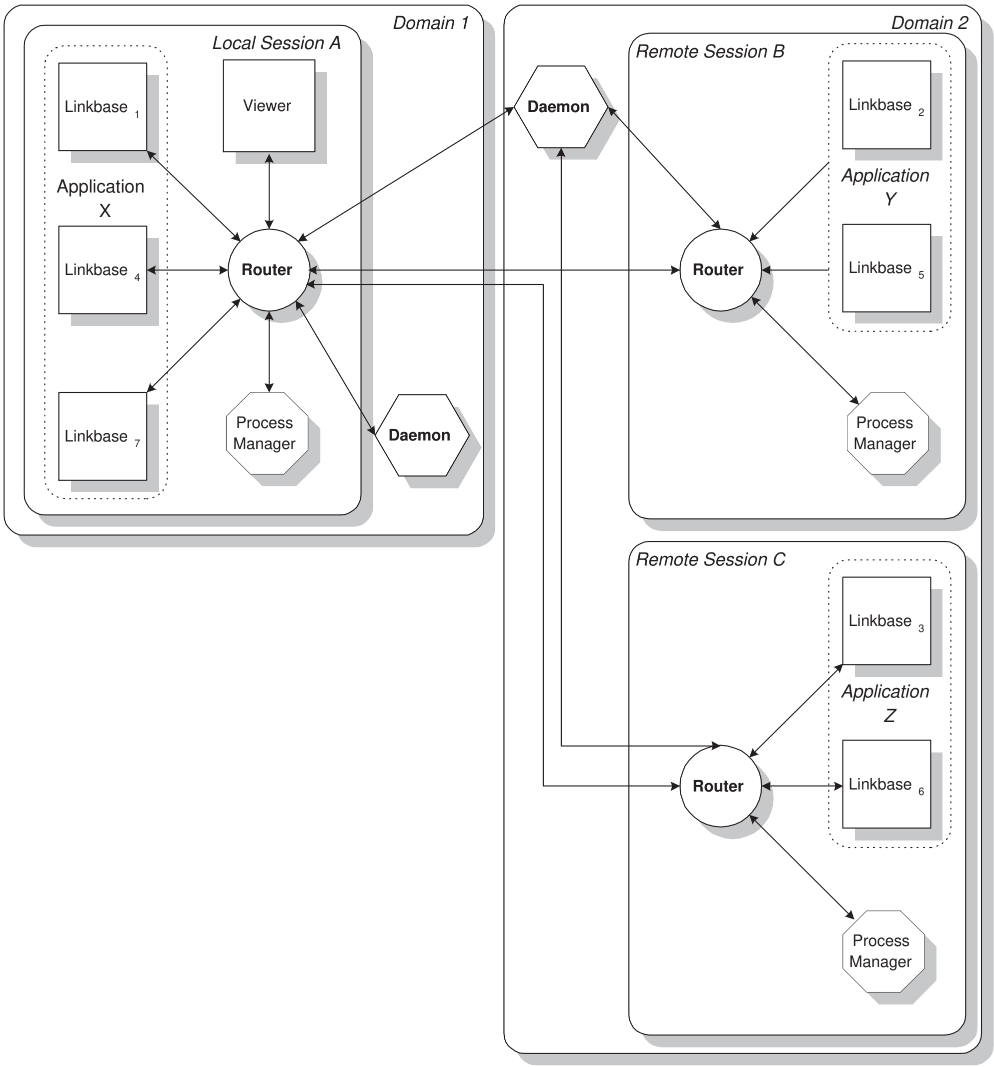


Figure 7.8: An alternative distributed architecture of Microcosm TNG.

Using Hill's extended version of Microcosm, figure 7.7 illustrates one possible system configuration for this scenario. To maintain the advantages of a serial filter chain, figure 7.7 depicts the circuitous route that a follow link request would be likely to take in the scenario. To reduce clutter, only the logical path is shown in the diagram, but the actual path necessitates many more steps, as explained in chapter 6.

Under Microcosm TNG, a solution to the same scenario can be observed in figure 7.8. This shows the pathway of an identical follow link request emanating from the viewer in Local Session A and arriving at the linkbases in Remote Sessions B and C.

This logical diagram has a very close mapping to the actual architecture.

At the expense of ordering, it is clear that as the number of linkbases in a remote session increases Microcosm TNG offers a more scalable solution. In addition, as distributed user sessions are supported each of the linkbases can execute on a separate machine. If, under TNG, either Remote Session B or C were to fail then those services would no longer be available, but under Microcosm the integrity of the chain would be jeopardised.

Both systems adopt a peer model and, with reference to point 5 of the requirements in section 7.2, are able to model the range of configurations recommended by Hill [110] for a versatile system.

7.8 Hypermedia Services

Previous sections in this chapter have described the refinement of the hypermedia application, the conceptual model of Microcosm TNG and the architecture to support application sharing between distributed user sessions. The conceptual model of Microcosm TNG comprises three primary facets and within the next three sections each facet will be described in some depth.

As described in section 7.4, a session can be considered as a container for multiple hypermedia applications. It is clear though that some processes exist independently of a hypermedia application, so a distinction between application specific services and hypermedia system services was introduced. Hypermedia system services represent the components that can be considered as infrastructure, whereas the application specific services represent the components that mediate access to application data, such as the documents and links. Each user has a set of hypermedia system services local to their session, whereas application specific services can be located in both local and remote sessions. Listed below is a table of the common components of Microcosm TNG, each placed in the appropriate category to illustrate the distinction.

Application	**System**
Linkbase	Viewer
Computed Links	Linker
Docuverse	Available Links

The components present in the table above were originally reviewed in chapter 4. They have been significantly redesigned to operate in a distributed setting and deliver the functionality of Microcosm TNG.

7.8.1 Linker

One of the problems outlined in section 7.3 pre-empted the development of an alternative mechanism with which authors can specify simply the linkbase in which a link should be created. In addition, it proved necessary to extend the linker to accommodate the new model in which multiple hypermedia applications can be present in a user session. These two issues were addressed simultaneously in the design of a new linker process.

The linker process declares an interest in receiving *start link* and *end link* messages by submitting entries to the message router similar to those listed below. As the linker is a hypermedia system service it registers using the nominal application *system*.

```
/64.128.6397/system/linker/64.8.2461/*/start.link
/64.128.6397/system/linker/64.8.2461/*/end.link
```

Upon receiving a pair of messages with *start link* and *end link* actions, the linker queries the message router to determine the active linkbases. An interface is then presented by the linker in which the user can supplement the anchor information provided. In addition, the user selects from a list the application linkbase(s) in which the link is to be created.

Once these details have been captured, the linker encapsulates the information into a *create link* message and sends it to the linkbase(s) specified by the user. Regardless of whether the linkbase(s) specified reside in local or remote sessions, the linker can appropriately target a message by formulating a loosley constrained destination address. The following address will match upon a process in any connected session registered under the hypermedia application *lions*. The file `lions.raw`, in which links are stored, is included in the address to distinguish the linkbase from other linkbases that may also be associated with the *lions* application, while *create link* indicates the service.

```
/*/lions/*/*/lions.raw/create.link
```

7.8.2 Linkbase

As described in section 4.7.2, a linkbase is responsible for creating new links, resolving follow link actions and returning all of the links within a given document. To advertise these services the linkbase process registers with the message router entries similar to those listed below. As a linkbase is an application specific service it registers using the name of the application.

```
/64.128.6397/lions/linkbase/64.8.3528/lions.raw/create.link
/64.128.6397/lions/linkbase/64.8.3528/lions.raw/follow.link
/64.128.6397/lions/linkbase/64.8.3528/lions.raw/find.buttons
```

It is possible with Microcosm TNG for a user to configure their session in such a way that multiple linkbases can be assigned to each run on a separate machine. An arrangement such as this allows a degree of parallel processing to be exploited. With reference to point 4 of the requirements in section 7.2, this illustrates how the distributed architecture of Microcosm TNG can provide a flexible and scalable solution.

When a linkbase receives a message with a *follow link* action it determines from the sender tag the session from which the message originated. The resolver function is then applied, parameterised by the source anchor information. If any matches are found a *dispatch* message is built containing destination document information. As the linkbase may not reside within the same session as the user's available links process, the session identifier extracted earlier is applied within the destination address to ensure that the message is returned to the correct user session.

```
/43.182.4663/*/*/*/*/dispatch
```

In Microcosm, each time the linkbase resolver function finds a match, a new message is created and sent along the filter chain. This approach has the advantage that the user is likely to observe some results within a short space of time with the remainder *trickling* in until the search has completed. Unfortunately, this approach does not scale well and hence required modification for use within a distributed setting. Instead, the linkbase has been re-engineered to build a compound message which contains all the matches for a given query. Although no immediate feedback is supplied to the user, this approach reduces communication overheads as only a single message is transmitted regardless of the number of matches found.

Whenever a new specific link is created, the linkbase responsible issues a message containing information about the new link to all processes that declared an interest. This allows viewer processes in connected sessions to update their displays dynamically and make a new link instantly available. This is an example of notification raised in point 1 of the criteria in section 7.2. Again, using a loosely constrained destination address a linkbase can target processes in connected sessions that are operating upon document number 20.03.19.45, in which the new source anchor has been defined, that wish to receive *found button* messages.

```
/*/*/*/*/20.03.19.45/found.button
```

7.8.3 Available Links

The available links filter was first described in section 4.7.6 as the component that presents the results of a link traversal or search action. For the available links process to receive messages regarding matched documents it submits to the message router a registration entry similar to the one below.

```
/64.128.6397/system/availlinks/64.8.4971/*/dispatch
```

An interface allows the user to peruse the list of matched destination documents. If the user selects a document to view, the available links process sends a message containing the document details to the process manager to launch the viewer.

```
/64.128.6397/*/*/*/*/launch
```

A simplified view of the mode of interaction with Microcosm is such that a link traversal or query is issued and resolved, one or more documents are selected from the available links filter, and the cycle is repeated. Each time the cycle is repeated the available links filter replaces the document list with a new one. Microcosm only allows queries to issued and resolved in a sequential manner. This is not a satisfactory solution for a distributed information system. When using search engines through the WWW it is common practise to open multiple browsers and conduct searches in parallel.

To overcome these limitations the infrastructure was enhanced and the available links process redesigned. The message router was enhanced to generate a unique message identifier composed of the session identifier and an incremental counter, e.g. 152.78.64.128.6397/43. Prior to sending a link traversal or search request a unique message identifier is appended to the message. As remote link services may respond to multiple and distributed requests the unique message identifier of the request is copied into the reply message to allow the response to be matched with the request. Upon receipt of a message the available links process checks the unique message identifier. If it is the first occasion that the available links process has noted this message identifier then a new window is created in which the results are presented. Alternatively, if this is not the first time that the message identifier has been noted, then, providing that the user has not terminated the request by cancelling this particular results window, the results are appended to the appropriate previously opened window. This can be appreciated in figure 7.9

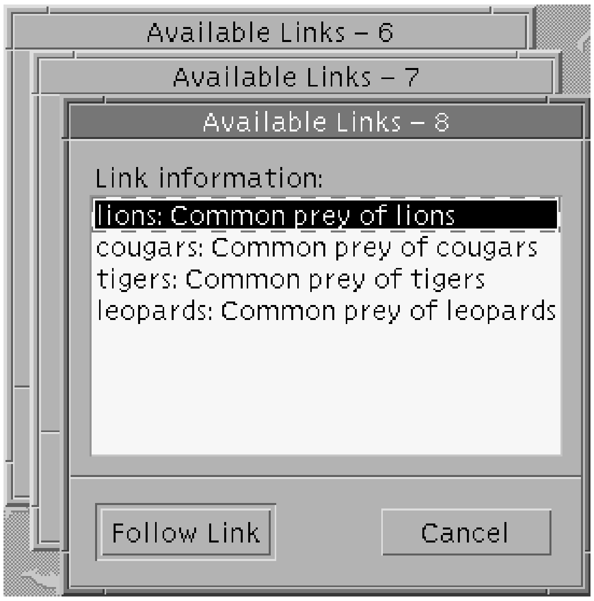


Figure 7.9: The Available Links process supporting multiple asynchronous queries.

These extensions provide support for multiple, asynchronous queries within Microcosm TNG, essential within a distributed system which can be subject to network delays. This feature also allows the user to retain useful lists of matched documents produced by the available links process, as opposed to them being erased each time.

7.9 Document Management Services

7.9.1 Is a Document Management System Necessary ?

Third party applications are designed to store and retrieve data from the native file system, and cannot usually be adapted, for example, to redirect these actions through a hyperbase. As such, it was considered important that the native file system should be used as the resource storage layer for Microcosm [102]. In keeping with the principle of allowing third party applications to continue accessing their data, Microcosm does not import document content. As MS-DOS places a restriction upon the length of filename and provides no means of attaching supplementary information to a file, a Document Management System (DMS) was introduced as an additional layer of indirection over the native file system. The DMS maintains a database of all the documents known to Microcosm together with any user-defined attributes regarding each file.

The introduction of a DMS represented a compromise to the open design of Microcosm. The additional layer between the file system and Microcosm meant that fully and partially aware viewers now access documents using an alternative suite of functions, while wrappers were required to preserve support for unaware viewers. As the DMS is not a true component of the native file system, it is not equipped to monitor and reflect any changes made to files, such as deletion, through the operating system interface. However, these disadvantages must be measured against the ability to read meaningful document names, establish new attributes, arrange documents into logical groupings and the provision of an alternative interface for browsing and querying a document collection.

7.9.2 Limitations of the Microcosm DMS

As alluded to in section 7.3, the characteristics of the communication model precluded the designers of Microcosm from allowing the DMS to be provided as yet another service in the filter chain. This problem was circumvented by embodying the DMS services in a shared library and allowing the system components to access them via an API.

While this decision provided a solution it also introduced an asymmetry to the design. The conceptual view of the system with a presentation front end and a link service back end was augmented with a DMS as an adjunct to both. A certain incongruence also exists in Microcosm between the way in which links and documents are managed. Nelson [144] coined the term *docuverse* to refer to the universe of documents, but in the context of Microcosm the term has been used to refer to the documents with respect to a single hypermedia application. An interface allows users to browse the documents in the docuverse. It soon became apparent that users often wished to make available in one application documents that were originally associated with another related application. This is not possible with the Microcosm DMS, as only one docuverse can be active at any one time. This limitation encouraged users to abandon a modular approach to document grouping in favour of importing every document into one large docuverse in order to make the documents available across all of their hypermedia applications.

Design modularity and potential for information reuse has led the architects of many open hypermedia systems to provide support for multiple linkbases. Although Microcosm supports multiple linkbases, the situation described above shows that similar scope for modularity is not extended to the management of documents. By taking a *macro* view, one may consider a docuverse to be the universe of documents with respect to a hypermedia application. By contrast, a *micro* view would focus on a smaller set, or microcosm, of the document collection. As such, a microcosm of the document collection can be termed a *docucosm*. The assimilation of multiple docucosms allows an improved modular construction of the universe of documents, or docuverse, to be achieved. This more versatile approach to document management more accurately mirrors the manner in which multiple linkbases are supported.

By redefining the notion of a hypermedia application and extending a session to accommodate multiple hypermedia applications, explained in section 7.4, the problem described above of making documents associated with one application available to that of another related application is eradicated.

The fundamental reason that the Microcosm DMS has these limitations is attributable to the fact that it is provided as a shared library rather than as an autonomous process, for reasons elaborated upon earlier. The majority of components in the Microcosm architecture are able to communicate by passing messages around the system, but because the Microcosm DMS is implemented as a shared library and disconnected from the central communication pathway it is unable to be proactive and issue messages to the rest of the system. In the requirements of distributed open hypermedia systems of section 7.2, it was argued in the first point that it is especially important for users of large distributed information spaces to be notified of relevant new information and changes to transient information. For this requirement to be met, a DMS must be capable of autonomy and able to communicate the appropriate event messages to the rest of the system in response to documents being imported, exported and updated.

The evolution of hypermedia systems traced in chapter 2 has led the community to converge upon the belief that an open and extensible link service is best provided by means of an open protocol. Although the design of the Microcosm link service adheres to this philosophy, it is clear that the design of the Microcosm DMS contravenes it. The limitations of the Microcosm DMS described have been rectified by representing it as an arbitrary number of *docucosm* processes capable of interacting with the hypermedia system using open protocols.

7.9.3 Distributed Document Management

The primary *raison d'etre* of a distributed DMS is to facilitate transparency. Location transparency is the capability of providing uniform access to documents regardless of whether they reside on a local or remote file system. Source transparency is the capability of providing uniform access to documents regardless of the type of the repository, e.g. FTP, HTTP.

7.9.3.1 Document Naming Schemes

A closely related issue of importance in the design of a distributed DMS is that of naming. The most prevalent global naming scheme is the Uniform Resource Locator (URL) used by the WWW. The URL provides a succinct and generic format for referencing any document across the Internet. Despite the fact that any alteration to the absolute components of a URL will result in a retrieval error, it has quickly become accepted as a *de facto* standard. While acutely aware of the volatility of the URL, Grønbæk *et al* [92] acknowledge the benefits: URLs are visible, human readable, able to exist independently of the hypermedia allowing them to be manipulated, copied and exchanged.

A plethora of schemes for improving upon the URL have been proposed and are currently under investigation by the Uniform Resource Identifier (URI) Working Group [176]. Most of the suggestions for providing stable Uniform Resource Names (URN) are based on a hierarchical name space with naming authorities responsible for resolving specific subtrees.

Hill [110] extended the Microcosm DMS to work in a distributed environment and adopted a naming scheme based on the syntax of the URL. Instead of a pathname after the machine name, the Microcosm unique document identifier was appended. The naming convention used in the Microcosm TNG distributed DMS extends this concept a level further. As each document is associated with a hypermedia application this information also forms part of the name. The syntax of a Microcosm TNG document identifier is:

```
<access type>://<hostname>/<application name>/<unique document id>
```

For Microcosm TNG documents the access type is `mcmtng`, but the adoption of the URL syntax eases integration with existing Internet services as described in section 7.9.3.3.

7.9.3.2 Microcosm TNG Distributed DMS

As stated in section 7.9.2, a Microcosm TNG docuverse is composed of one or more docucosm processes. Although a docucosm is functionally similar to a Microcosm docuverse, interaction with the rest of the system is distinctly different. Under Microcosm TNG a viewer is invoked with a document identifier as a parameter. The viewer can request additional information about the document by sending a message request to the application specific docucosm(s), whether local or remote, quoting the document identifier. The appropriate docucosm will then respond with a message containing the entry in the document database. The viewer can issue a request to that docucosm to stream the document over the network to it. Upon receipt of such a request, the docucosm spawns a new thread and opens a network connection. It then sends a message reply to the viewer stating the host and port number of the connection over which the document can be read. The viewer can then connect to the docucosm thread and retrieve the document.

As the DMS serves as a layer of indirection, it is flexible enough to allow a document identifier to reference remote sources of information. In this manner, for example, it is possible for a document identifier to reference a server transmitting audio, video, or even a multicast feed issuing the latest stock prices. Instead of spawning a thread to manage the retrieval of a document, in such cases the docucosm will simply return the location and the protocol required for communicating with these alternative information sources.

As discussed in section 7.9.1, many third party applications are designed to read documents from the local file system, not streamed over the network. To meet this need, a generic wrapper process was developed that spools a file locally from a docucosm and manages a third party application.

As mentioned in 7.8.1, links can be created between documents in different hypermedia applications regardless of whether the hypermedia applications are local or remote. If a user has established in advance connections to any referenced remote hypermedia applications, using the technique described in section 7.7.2.1, then the retrieval of remote documents will occur transparently in the way described above. But the case must also be managed whereby a link is followed that resolves to a document from a remote hypermedia application to which the user has not connected in advance.

A *proxy docuverse* process has been developed that monitors a user session and keeps track of the hypermedia applications in use. The proxy docuverse registers to receive the same message events as a regular docucosm process, but by specifying a wildcard in the registration entry it receives messages relating to *all* hypermedia applications. Whenever a viewer issues a request for a document the proxy docuverse also receives this message and checks whether the hypermedia application is present within the user session. If the hypermedia application is present within the user session then the message is ignored by the proxy docuverse, but if it is absent then the proxy docuverse issues notification asking whether the user wishes to connect to the remote session from which the application was originally published. If the user sanctions the opening of the remote hypermedia application then it is performed transparently, otherwise the required document is retrieved.

The proxy docuverse is able to determine the location of the remote hypermedia application from the Microcosm TNG document identifier, introduced in section 7.9.3.1. As described in 7.7.1, a Microcosm TNG session can comprise a number of processes distributed over a network of machines. As such, it is unlikely that a docucosm process will always execute on the same host. It is for this reason that the host address quoted as part of the document identifier is that of a domain daemon, and not the address of the host upon which the docucosm process was executing at the time the document was imported.

The desire of Haan *et al* [96] for a link traversal that can cause another web to be opened has thus been realised and extended to facilitate a link traversal that can cause another hypermedia application to be opened.

As the processes that comprise the Microcosm TNG DMS are able to communicate with the rest of the hypermedia system, it is now possible for the presence of new and relevant documents to be announced. This allows a user to request notification of specific new documents, for example, by a particular author, about a certain hypermedia application or containing specific keywords. This illustrates how the Microcosm TNG DMS can become proactive, with potential for aiding users of large distributed information systems as mentioned in point 1 of the requirements in section 7.2.

7.9.3.3 Integrating Existing Internet Services

In point 3 of the requirements in section 7.2, the importance of integrating with external Internet information services was emphasised. The WWW is able to access documents from FTP, NNTP and Gopher servers, while Hyper-G can integrate with HTTP and Gopher servers. Hill [110] also describes how his distributed version of Microcosm might integrate with Internet services.

To provide viewers with consistent access to distributed hypermedia documents, the designers of Hyper-G and Microcosm are able to conceal the complexities of multi-protocol communication within the cache and DMS respectively. This approach also has the advantage that support for future protocols can be added transparently.

Rather than re-programming a cache or DMS to provide explicit support for a new protocol, a more modular and dynamic approach to Internet service integration was taken with Microcosm TNG. When a docucosm process receives a request for a document to be streamed it first parses the access type of the document identifier. If the access type is not `mcmtng`, then the docucosm process checks whether there is a *gateway* process registered that is able to service this protocol request. If so, the docucosm process forwards the message request to the gateway process. Upon completion of the request the gateway process sends the docucosm process a message stating the location to which requested document has been spooled. The docucosm can then stream the document to the viewer in the familiar manner.

FTP, HTTP and Gopher gateway protocol processes were developed for Microcosm TNG. Rather than implementing full support for the three different protocols within each gateway process, a shortcut was taken that involved exploiting WWW proxies. The role of a WWW proxy is to accept a URL and retrieve the document. As the WWW supports FTP, HTTP and Gopher, a WWW proxy can be used to retrieve documents from these Internet services. Each of the Microcosm TNG gateway processes simply connect to a specified WWW proxy, issue a URL and retrieve the document from the WWW proxy over a network stream. This specific use of WWW proxies is obviously limited to the protocols supported by the WWW, but the notion of gateway processes to serve the DMS, but able to exist independently, is a modular, dynamic and extensible approach to integrating with Internet services.

7.9.3.4 Caching

The WWW implements caching within the browsers and proxies; Hyper-G implements a cache at the client; Hill [110] facilitates caching within the DMS in his distributed version of Microcosm. Rather than prescribing a specific solution for caching in Microcosm TNG, a variety of schemes can be accommodated.

No cache When browsing local documents it can prove more efficient to suppress caching. It is normal to disable caching during the streaming of continuous media.

Per-viewer cache As with WWW browsers, it is possible for all caching to be delegated to the individual viewer. While this approach has its merits, it may not be possible for such a cache to be shared between the different viewers of a distributed hypermedia system.

Per-session cache This approach is where all document requests are sent to the proxy docuverse process which retrieves and caches the document before streaming it to the viewer. This approach solves the previous limitation, but the user may also wish to share the contents of their cache with other users in the local domain.

Per-domain cache A session may be instantiated with a proxy docuverse process that is published as a service, allowing connections to be made to this session to make use of the service. In this manner a proxy docuverse can be made available as a domain wide caching service.

It would be possible to use each type of cache in isolation, or as part of a three tiered caching scheme.

7.9.3.5 Browsing Docuverses

The Microcosm DMS, described in section 4.9, provides an interface similar to that of a file manager allowing a user to browse the logical structure of a hypermedia application and the documents therein. The DMS interface is not provided as a separate utility that makes use of the DMS, but instead is embedded within the shared library. By separating the process from the interface, developers can provide alternative views of the document space, such as those described by Weal [178].

The WWW provides no explicit support for the browsing of remote HTML document hierarchies; Hyper-G supports the browsing of remote logical collections; no provision was made for browsing a remote DMS with Hill's distributed version of Microcosm. The docucosm process was divorced from the interface and the resultant interface for browsing the logical structure of Microcosm TNG hypermedia applications is presented in section 7.10.3.

7.10 Presentation Services

7.10.1 What the User has to Face

The definition of a hypermedia application, described in section 7.4, subtly changes the emphasis of a user session. A Microcosm user session is bound to the context supplied by a single hypermedia application. Whereas in Microcosm TNG the session becomes the context; the fusion of the hypermedia applications present.

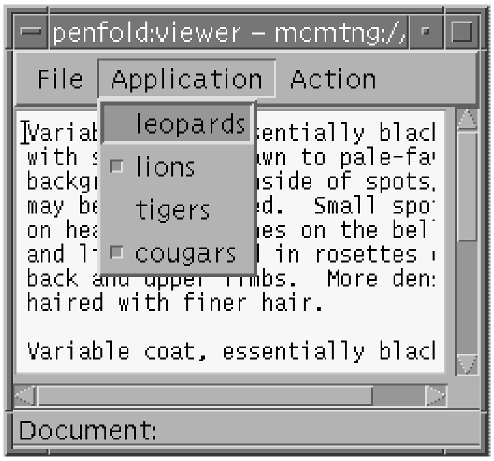


Figure 7.10: A viewer showing two selected applications in the menu.

It was originally envisaged that a user would incorporate a collection of related hypermedia applications within their session and that each action would be applied across *all* applications. The granularity of this approach proved too coarse, as it precluded a user from applying a particular action to a selective subset of the applications present within the session. The viewers were subsequently modified to determine the hypermedia applications within the session dynamically and provide an additional menu listing them. Toggle buttons on each of the application menu items allows the user to specify precisely the application(s) to be targeted with any subsequent action. The simple text viewer in figure 7.10 shows that the *lions* and *cougars* hypermedia applications are currently selected as the recipients of the next action.

Although this interface provides a simple and intuitive extension to the *selection/action* metaphor of Microcosm, it can add another stage when interacting with the system to formulate an action. To combat this a new utility was developed called the *tngtool*. This utility provides two floating palettes listing the available hypermedia applications and services. The user can create a custom *shortcut* by dragging from the palettes and dropping onto the shortcut dialog the required application(s) and service(s). The user may then drag a selection from a viewer and drop it onto the shortcut whereupon the request will be formulated and sent. The shortcut dialog interface seen in figure 7.11 is currently limited to text only selections.

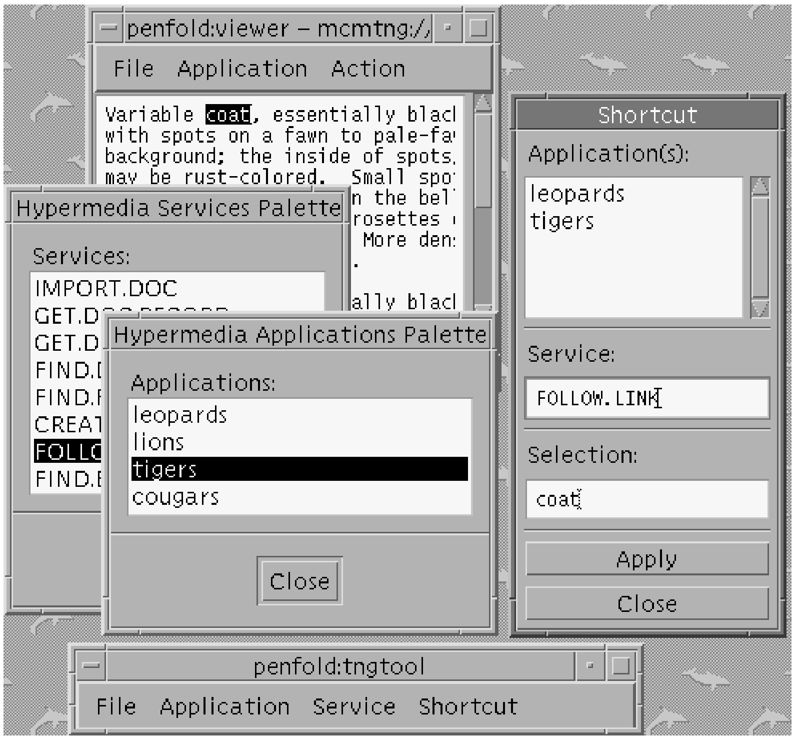


Figure 7.11: The construction of a shortcut using the tngtool.

7.10.2 Viewer Integration

The increasing importance of integrating with third party applications hosted on a disparate range of operating systems was the third point raised in the requirements of section 7.2. To examine whether Microcosm TNG was sufficiently open to meet this requirement two exemplar third party applications were selected: Emacs on UNIX and Microsoft Word on Microsoft Windows.

Written in Microsoft Visual Basic for Applications, a program was developed that is loaded by Microsoft Word and executes transparently to the user. The program augments the Word menus with the Microcosm TNG menus described above. An API to the HCM has been ported to Microsoft Windows and provided as a Dynamic Link Library (DLL). Visual Basic for Applications is able to call functions in a DLL. It proved a trivial exercise to extract the selection and offset from a Microsoft Word document, build a message to encapsulate this information and post it to the link service upon a menu action event.

As Emacs provides a powerful Lisp programming environment with which the editor can be customised and extended, an alternative integration strategy was adopted. In a similar fashion to the above integration, a Lisp program was developed that is loaded upon the start up of Emacs which executes transparently to the user and attaches additional menus. In contrast to the approach above, Lisp code was written allowing messages to be communicated over TCP/IP sockets to the message router and the rest of Microcosm TNG.

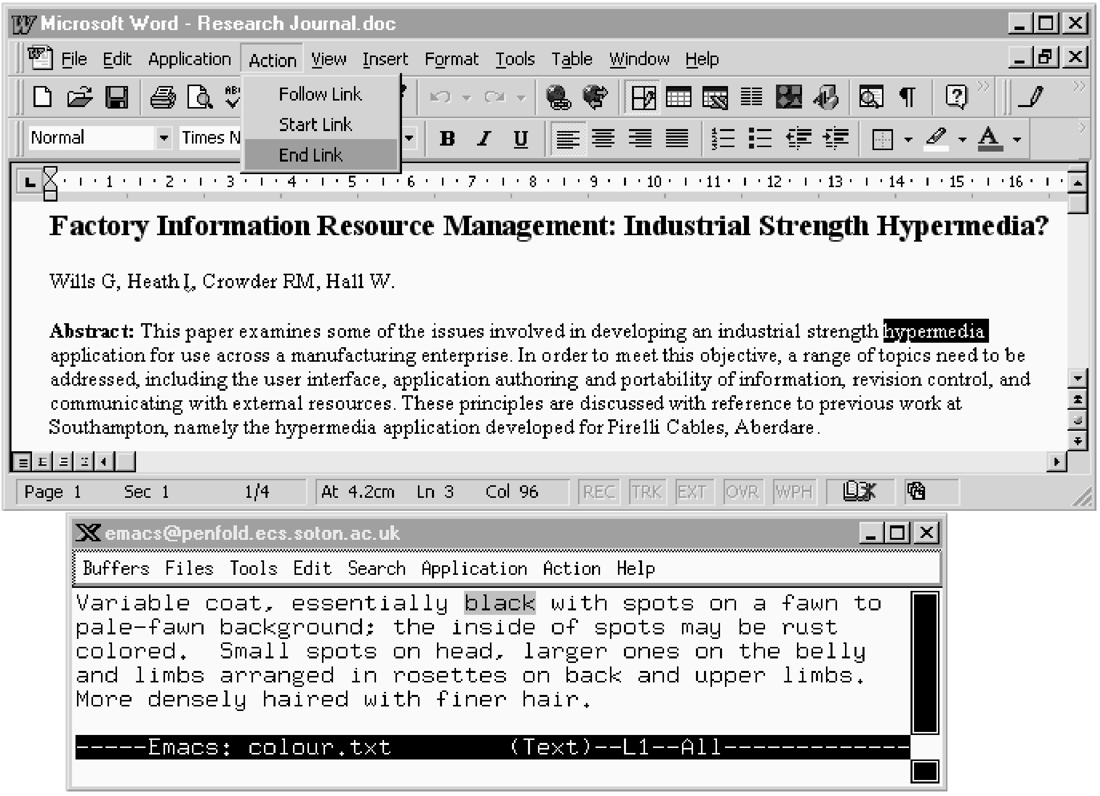


Figure 7.12: Creating a link under Microcosm TNG using Microsoft Word and Emacs.

This experiment demonstrates how, arguably, two of the most widely used applications on UNIX and Microsoft Windows can be integrated with Microcosm TNG with relatively little effort. Both applications provide very powerful programming languages which greatly aided the task of integration. Figure 7.12 shows a screenshot of a link being created between an anchor in Microsoft Word and an anchor in Emacs.

7.10.3 Heterogeneous User Interfaces

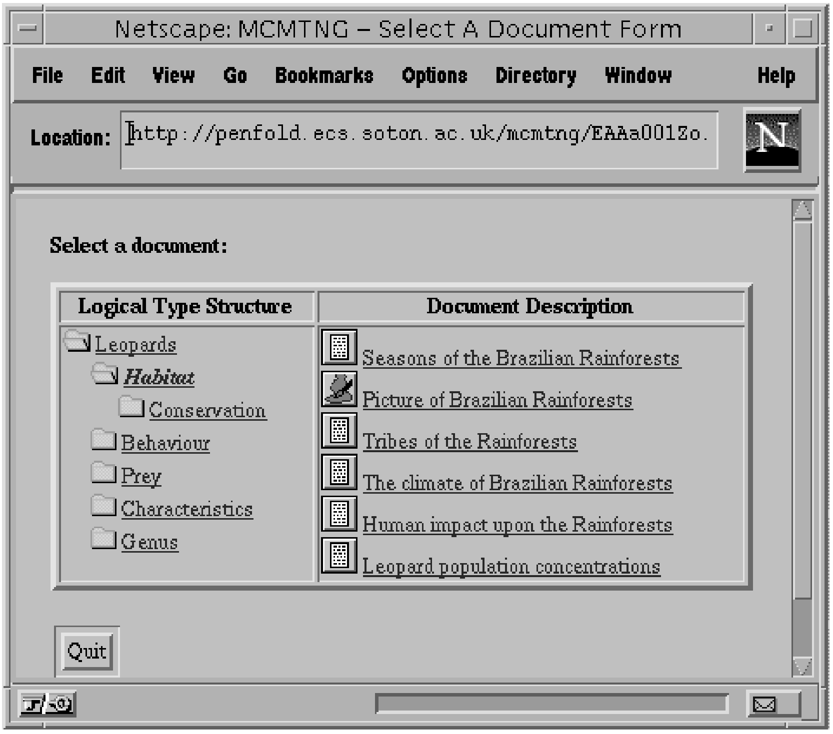


Figure 7.13: The user interface to the Microcosm TNG distributed DMS browser.

The case for divorcing the processing from the interface was introduced in section 7.9.3.5, but the argument is further strengthened when placed in the context of a distributed system. One of the guiding principles of distributed processing is that the processing should be performed on the most appropriate machine. X Windows alleviates this problem by allowing a process to execute on one machine but have its corresponding user interface redirected over a network to the display of another machine. Although X Windows can, in theory, allow the display of Microsoft and Macintosh applications to be redirected to another machine, unfortunately this technology is not supported as standard by these operating system GUIs. A more heterogeneous solution is required to allow user interfaces to be redirected to the display of another machine regardless of the platform on which it is executing.

One solution to this problem has been developed by Dale [47]. He advises separating a traditional process into two distinct but communicating processes, called the Process Handler (PH) and the Interface Handler (IH). The PH is responsible for dynamically constructing a new user interface. The interface is then communicated to the IH which arranges for it to be displayed. HTML is a prevalent GUI independent display language, and hence was selected as the language with which the PH specifies its user interface. When the interface is to be displayed, the IH requests a WWW browser to be invoked on the user's machine to render the HTML. The interface is an HTML form, and hence facilitates interactivity. When the user interacts with the form a CGI script is executed to relay the contents back to the IH and onto the PH for processing. The PH processes the contents of the form and generates a new interface, and the cycle continues.

This technology was used as the basis for a number of experimental user interfaces for Microcosm TNG such as a linker process, a text viewer process and an available links process. An interactive browser to the Microcosm TNG DMS was also developed using this approach. A screenshot of the browser can be seen in figure 7.13 showing the logical structure of the document collection in a hypermedia application about *Leopards*.

7.11 Summary

This chapter began by revisiting the six point definition of open hypermedia by Davis [48] and examining each point with increased emphasis upon distribution. Distilled from this activity and the review in chapter 5 were the three primary requirements of distributed information sharing, distributed information organisation and extensible distributed services. These core requirements drove the design of Microcosm TNG, an open and extensible distributed hypermedia system that drew heavily upon the experiences from the Microcosm project.

The notion of a hypermedia application was redefined to be a binding of a logical grouping of documents, the services that operate on the collection and the associated configuration information. This specification provides a higher level of abstraction and allows each hypermedia application to be considered as a self-contained, publishable and shareable entity. Familiar Microcosm hypermedia applications including Caerdroia and Cell Biology [102] were ported and trialed under Microcosm TNG.

It has been described how the Microcosm TNG infrastructure, built above the HCM, can support distributed user sessions. A mechanism for browsing and connecting to hypermedia applications hosted in remote domains was also reported. Once a connection has been established to one or more remote hypermedia applications their link services are accessed in a transparent manner.

The three primary facets of the open hypermedia system are each described in turn. A number of modifications to various link service components were outlined which proved necessary in order for the system to be suitable for distribution. The Microcosm Document Management System was analysed and an alternative architecture suggested which has implications for distributed operation, naming schemes, integration and caching. Finally, the front end presentation services were described and the impact that a distributed infrastructure has upon this facet. An experiment to demonstrate cross-platform linking was also described that involved the integration of Emacs on UNIX and Microsoft Word on Microsoft Windows with Microcosm TNG.

It has been described how the three core requirements for Microcosm TNG have been satisfied through the notion of a hypermedia application:

distributed information sharing Hypermedia applications can be published granting access to the collection of documents therein.

distributed information organisation The logical structure of a distributed document collection of a hypermedia application can be browsed using the distributed DMS browser.

extensible distributed services A range of supporting link services can also be supplied as part of the published application.

Although Microcosm TNG addresses many of the issues raised in section 7.2, it is hoped that the open framework can provide a workbench with which to explore the unresolved issues. The next chapter describes the reflexive model that enables authors to use existing distributed hypermedia applications as building blocks with which to create rich hierarchies of information. This mechanism offers potential for the reuse of information at a higher level.

8 A Reflexive Model for the Construction of Hypermedia Applications

8.1 Introduction

The design and architecture of a distributed, open hypermedia system that allows users to publish and share their hypermedia applications was described in chapter 7. This chapter begins with an examination of the internal structure of a hypermedia application and discusses measures that may improve maintenance and reuse. The notion of a hypermedia application is explored further and architectural extensions to Microcosm TNG are described which facilitate the dynamic aggregation of hypermedia applications.

8.2 The Resource Based Approach to Hypermedia

When summarising Microcosm, Davis [48] describes it as:

> ...an open hypermedia system which lends itself to a *resource based approach* to the collection and navigation of materials.

The *resource based approach* alluded to above involves effectively two separate but concurrent tasks which occur during the authoring process:

- building a resource base
- authoring within a resource base

The first phase entails gathering numerous entities of information that are potentially useful to the application area and transforming them into their digital equivalent to form a collection of multimedia assets. These digital information resources are often regarded as assets as they offer the potential to be repurposed for a different context. Once these assets have been organised into a logical structure, the second phase of creating links between them can begin. By providing independent links between the information resources different sets of links can be provided for different groups of people for different purposes.

It is important to note that whereas previously the content has been defined completely prior to any programming or authoring, the design process is now an ongoing process with new links and content areas only becoming apparent during the process of authoring.

Hall *et al* [102] offer more specific advice regarding successful techniques for both phases. Davis [49] argues that the resource based approach can aid with information reuse, allowing hypermedia applications to be reconstructed and extended without incurring the usual economic penalties that embedded links imply. This practice is not limited to hypermedia application development with Microcosm, but has also been applied to the WWW [54]. How these issues can be managed with Microcosm TNG is discussed in the next section.

8.3 Structuring Hypermedia Applications

Although a hypermedia application has been explicitly defined, significant flexibility exists within the realm of this definition to publish a hypermedia application in a variety of ways. A number of approaches have been proposed for the conceptual design of hypermedia applications, including HDM [72], OOHDM [164] and RMM [116]. No attempt is made here to contribute to this knowledge, but some pragmatic suggestions are offered as to how hypermedia applications under Microcosm TNG can be organised to address the issues of maintenance and reuse.

Coupling and cohesion are two concepts borrowed from software engineering. In the software engineering domain, coupling refers to the extent and complexity of the interconnections between individual software modules, whereas cohesion refers to the extent of self-consistency and clarity within modules. Minimal coupling means that changes have minimal effects upon other components. This results in modules which can be more readily modified and reused. A high level of cohesion can be attained if a module is well defined and understood. This allows the implications of changes to be appreciated more easily.

It is useful to examine the concepts of coupling and cohesion in the context of hypermedia. Lowe *et al* [134] relate the notion of cohesion to be:

> ...the process of chunking or breaking information into nodes. If the chunking process is done well (i.e. the nodes are cohesive) then we have both a high level of reusability and an improved understanding of each node, and hence the implications of any changes.

In software engineering the aim is to reduce coupling, yet in hypermedia the notion of coupling, or associative linking, lies at the heart of the technology. Lowe *et al* suggest that:

> ...we wish to *control* rather than minimise coupling. If a given region of an information space is highly coupled to another region, then it will be difficult to make changes to the first without affecting the second.

The management of cohesion is largely the responsibility of the hypermedia author(s), but some suggestions are offered in the following sections as to how coupling can be *controlled* using Microcosm TNG.

8.3.1 Links

Abstracting links from documents allows the potential for information to be repurposed. Although only a single hyperweb is provided for each user of Hyper-G, systems such as SP3, DHM and Chimera support multiple hyperwebs. HyperDisco workspaces bear similarities to Microcosm TNG hypermedia applications except that a workspace supports only a single hyperweb, or context, and provides a fixed number of hypermedia services. HyperDisco, however, now supports inter-workspace links. By comparison, hypermedia applications under Microcosm TNG can have multiple hyperwebs (linkbases), extensible link services and support inter-hypermedia application links.

Through independent links between information resources, DeRoure [54] describes how different sets of links can be provided for different groups of people for different purposes at different times. This feature allows authors to partition links into logical clusters according to context. Extensions to the Document Management System, described in section 7.9, now also allow authors to partition documents (or more precisely document references) into clusters according to context. Brown [31] emphasises the importance of modularity. He argues that by splitting a hypermedia application into separate, largely independent modules, the task of maintenance can be greatly eased.

On solutions to reducing coupling (controlling hypermedia links) between separately managed sites, Lowe *et al* suggest:

> ...constraining the coupling between regions to well defined avenues of navigation. One mechanism for achieving this would be to ensure that each region of the information space has a very limited number of well defined entry points.

While the recommendation by Lowe *et al* is indisputably sound advice, such prescriptive measures may hamper the rich interconnectivity required for flexible browsing. Rather than banning rich interconnectivity, the approach advocated here is to organise the link storage along the dimensions of context and location. Figure 8.1 conceptualises the anatomy of a modular hypermedia application within Microcosm TNG. Each row represents a context, or theme, that the author focuses upon in the hypermedia application. The documents that have primary relevance to a context are located on the same row. Although links pertinent to a context are also located on the same row, the specific linkbase in which they are stored is dependent upon the location of the destination document. The columns in figure 8.1 illustrate a sliding scale of location, from intra-context links through to inter-hypermedia application links.

In addition to specific point-to-point links, generic links and computed links can provide hitherto unknown points of access into applications. As such, generic and computed links also have the potential to increase cohesion but without incurring the penalty of additional coupling. Carr *et al* [39] argue that generic links and specific links offer complementary services by allowing an author to create a set of links that can be used to *come to* documents as well as a set of links which *go to* other documents from the current document.

Although Microcosm TNG does not force an author to modularise their links at

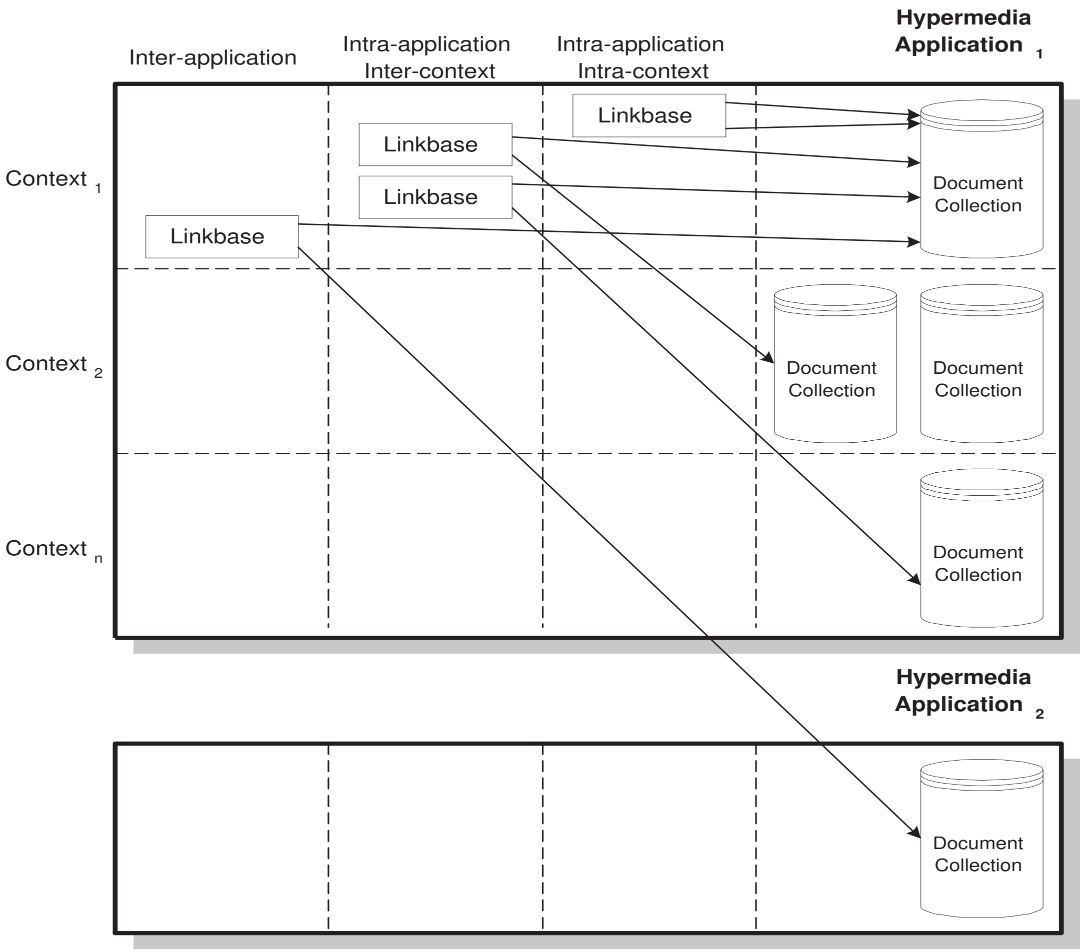


Figure 8.1: Organising link storage along the dimensions of context and location.

all, organising links along the dimensions of context and location has advantages. It makes obvious the degree of coupling between contexts and applications and allows this coupling to channelled through specific linkbases. As Brown [31] recognises, a modular approach to the organisation of a hypermedia application is of benefit for maintenance and reuse. If inter-application links continually result in dangling links then the task of maintenance is significantly reduced as the linkbase containing the offending links can be swiftly identified. This approach also offers the potential for increased reuse as cohesion can be high and coupling well controlled. This organisation allows authors to reuse specific components from an application more easily.

8.3.2 Link Services

The most common usage is likely to take the form of an author publishing a hypermedia application comprising a collection of documents and a set of links, possibly also including a search engine and index. A remote user stands a better chance of being able to reuse in their entirety smaller hypermedia applications on a well defined topic than large applications covering a wide subject. That does not make large applications worthless from the perspective of reuse. A large hypermedia application can be published in a number of forms in order to expose a number of more specific views of the functionality. For example, single contexts can be exposed to offer a more focused view on a subset of the information.

For example, an author may choose to make available only the collection of documents and metadata belonging to a specific hypermedia application. This can be achieved by publishing a hypermedia application simply containing one or more docucosm processes to mediate access to the document sets. This, in essence, would be a service bearing similarities to that of FTP.

8.4 Should Dexter be Extended ?

The Dexter Model has received criticism for only focusing on the data model and not considering processes. Leggett *et al* [126] claim that a complete hypermedia reference model should address the aspects of both data and processing. A composite, or aggregation of documents, is the largest entity specified by the Dexter Model. Again, a composite is only concerned with data.

Although a composite is the largest entity specified by the Dexter Model, the term *hypermedia application* has been referred to frequently in the literature by Garzotto [72, 70, 71], Rossi [161], Balasubramanian [14], Davis [50], Hall [101, 102], Lowe [134] and Haake [94, 95]. Even though no formal definition for a hypermedia application exists these authors all use and refer to the term as some sort of meta-entity.

A formal definition of a hypermedia application was introduced in section 7.4 which allows instances to be considered as first class entities within the conceptual framework. This new meta-entity addresses one shortcoming of the Dexter Model by encompassing both the data and process facets. In the next section the notion of a hypermedia application is explored further. Architectural extensions to Microcosm TNG are described which expedite the aggregation of hypermedia applications themselves.

While we recognise that *application* is a heavily overloaded term in the computing vocabulary, the nomenclature has been inherited from Microcosm. Confusion has at times arisen when describing these ideas due to conflicting interpretations of an application. Perhaps it is appropriate to coin a term to replace that of a hypermedia application.

8.5 The Reflexive Model

8.5.1 Information Reuse

Davis *et al* [49] argue that the time required to create computer assisted learning (CAL) material has limited its introduction into mainstream education. Typically 100-150 hours have been cited as the time required to produce one hour of instructional material. Hall *et al* [102] estimate that an author can expect to dedicate as much as 70% of his or her time simply acquiring a digital representation of the original material.

Garzotto *et al* [70] identify two distinct reasons for the reuse of information. The first is for economic reasons and they estimate that with systematic reuse an application can cost around 20% of a regular application. The second reason is to present the information in a different context.

The initial phase of building a resource base of individual multimedia documents can prove to be a time consuming process. But if, when mining for relevant information, the user unearths an existing hypermedia application comprising related documents and services this could provide a rich seam of information immediately. By creating and publishing modular hypermedia applications with a well-defined scope, it is possible to raise the level of granularity with which new entities can be reused. To address this point, the following section describes how the notion of a hypermedia application has been further refined in order that an application may reference one or more existing applications.

8.5.2 Reflexive Hypermedia Applications

The definition of a hypermedia application was originally introduced in section 7.4. This definition provides a convenient abstraction, allowing each hypermedia application to be considered a self-contained, publishable and shareable entity. This approach allows entities to be associated with a hypermedia application that possesses context, while able to retain the desirable properties that exist at the document level. It has been shown throughout chapter 7 how users of Microcosm TNG can include both local and remote published hypermedia applications within their respective sessions to construct richer logical information spaces.

Abstraction is one popular technique that humans use for managing complexity. The abstraction of a hypermedia application allows users to construct sessions using higher level, contextual building blocks. The natural progression was to devise a mechanism for enabling authors to create new hypermedia applications from existing hypermedia applications [78]. This has been achieved by extending the definition of a hypermedia application to include one or more optional applications. This recursive definition ascribes to applications a reflexive property, hence the name of the approach.

This extension allows users to make references to other applications and dynamically associate them with their own structures to build information resources which may seamlessly comprise a number of existing distributed hypermedia applications [82, 79]. The three primary requirements of distributed information access, distributed information sharing and extensible distributed services supported by Microcosm TNG remain intact as a result of this evolution of the hypermedia application.

An example of the reflexive model in context is given in figure 8.2. *Encyclopaedia* is a hypermedia application comprised of many other hypermedia applications which describe its behaviour, both in terms of information content and functionality. Such an application is a dynamically extensible entity within which new applications can be plugged or unplugged at any time. As the application is not monolithic, or statically defined, alternative views can be generated simply by including or excluding various application sub-hierarchies. For example, a *Junior Encyclopaedia* may use some of the same applications as *Encyclopaedia*, but not incorporate others deemed unsuitable for younger audiences.

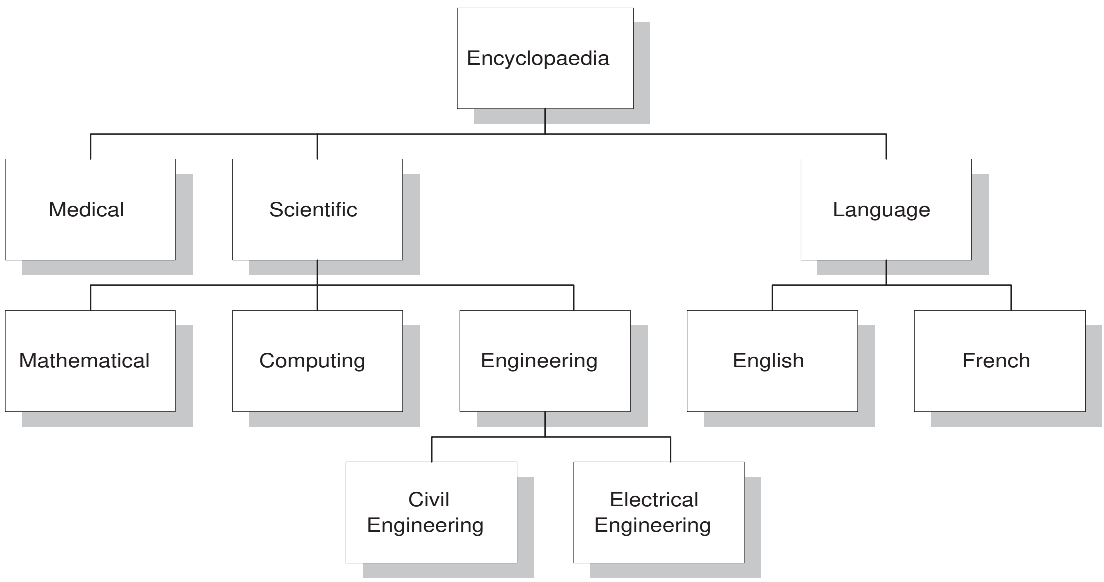


Figure 8.2: A hierarchy of reflexive Microcosm TNG hypermedia applications.

Lowe *et al* [134] define information to be the interpretation of data within a context. The reflexive model provides a flexible and extensible approach that allows a user to create multiple and different views on a set of hypermedia applications, where the context is dictated by the positions of the applications in the hierarchical arrangement.

By allowing hierarchical organisation of hypermedia applications the reflexive model provides the infrastructure for raising the level at which hypermedia resources may be reused. The resource based approach described in section 8.2 can also be extended to consider entire hypermedia applications, or indeed application hierarchies, as rich information resources available for reuse.

Ted Nelson, during his opening keynote address at the IEEE international multimedia conference in 1996, restated his long held belief that hypertext links to information over a network should be by reference only. He refers to this as transclusion and presented his transcopyright mechanism for engaging information providers in a contractual obligation to publish information in a stable form for a minimum time period. Although the reflexive model includes services in addition to documents, Nelson's principle of transclusion is still adhered to.

8.6 Architecture to Support Reflexive Hypermedia Applications

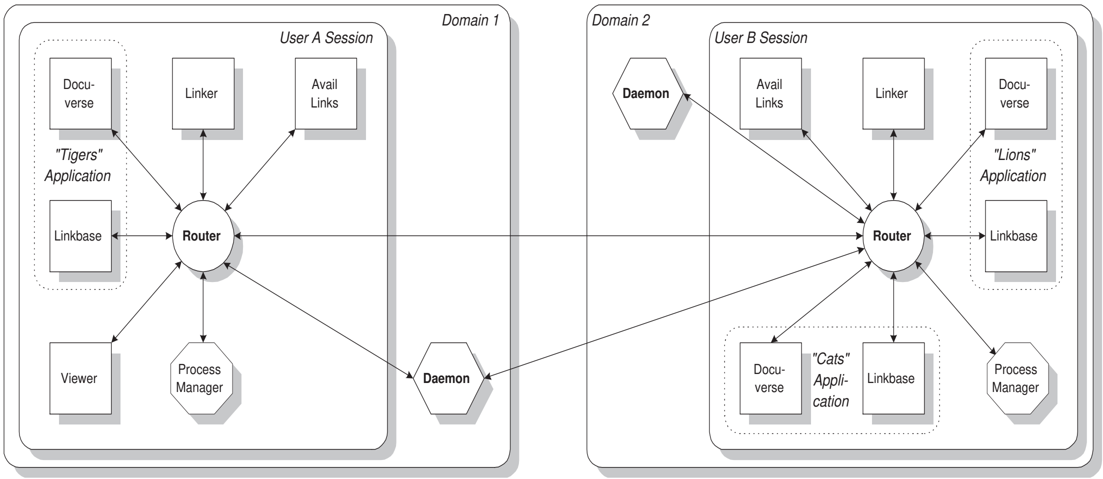


Figure 8.3: The *Cats* hypermedia application is enriched by the *Lions* and *Tigers* applications.

From figure 8.3 it can be observed that User A in domain 1 has several processes running within his/her session, as does User B in domain 2. The dotted lines denote collections of processes which have been published as hypermedia applications. User A has published an application named *Tigers*. With the *app browser* utility User B has forged a connection to User A's session in order to use the *Tigers* application to complement his/her application about *Lions*. User B then uses these two applications to enrich his/her more general hypermedia application on *Cats*.

The architecture of Microcosm TNG required some minimal extensions to support the interaction between distributed and reflexive hypermedia applications. These extensions and the issues raised are described in the following sections.

8.6.1 Hypermedia Applications as Process Groups

In section 6.4, it was explained how the structure of processes can be organised into peer, hierarchical and overlapping groups. By adhering to the custom template in section 7.6.4, it has been shown in chapter 7 how Microcosm TNG processes are able to register with a message router using hypermedia application names as process group identifiers. This allows messages to be sent to all group members. This is an example of the way in which the HCM supports peer group organisation.

Although the Microcosm TNG infrastructure provides a peer process model, the reflexive model allows the user to compose collections of processes dynamically into coherent and logically related hierarchies. The reflexive property of the hypermedia applications is modelled in the framework through the passing of messages between processes organised in overlapping and hierarchical groups.

8.6.2 Reflector Process

An additional *reflector* process was developed which enables hypermedia applications to be arranged in hierarchies. The process accepts as arguments the name of the hypermedia application to which it will belong followed by an arbitrary number of hypermedia application names that are to form logical children in the hierarchy. The reflector requires no user interface and can exist as a transparent application specific process.

This approach makes it trivial for User B from the scenario described in section 8.6 to construct a small application hierarchy with *Cats* at the root and *Lions* and *Tigers* as logical children. A reflector process needs to be started as part of the *Cats* application that accepts as parameters *Lions* and *Tigers*. The reflector process ensures that each service request to the *Cats* application is forwarded to both the *Lions* and *Tigers* applications.

In this scenario it is possible to appreciate how this principle is extensible and can be extrapolated to form larger distributed application hierarchies if desired. For example, User A may have constructed the *Tigers* application from two further applications about the *Siberian* and *Bengal* species of tiger. The constitution of the *Tigers* application is irrelevant to User B and is abstracted by the Microcosm TNG framework. Figure 8.4 shows one possible distribution of these hypermedia applications. Again, the inclusion of a reflector process within the *Tigers* application ensures that all service requests are forwarded transparently to both the *Siberian* and *Bengal* applications.

It is possible to draw an analogy between reflexive hypermedia application hierarchies and UNIX filing systems. UNIX allows physical storage devices to be *mounted* at any position within the hierarchy to create a seamless logical file system structure. The user can navigate around the file system without being concerned with which

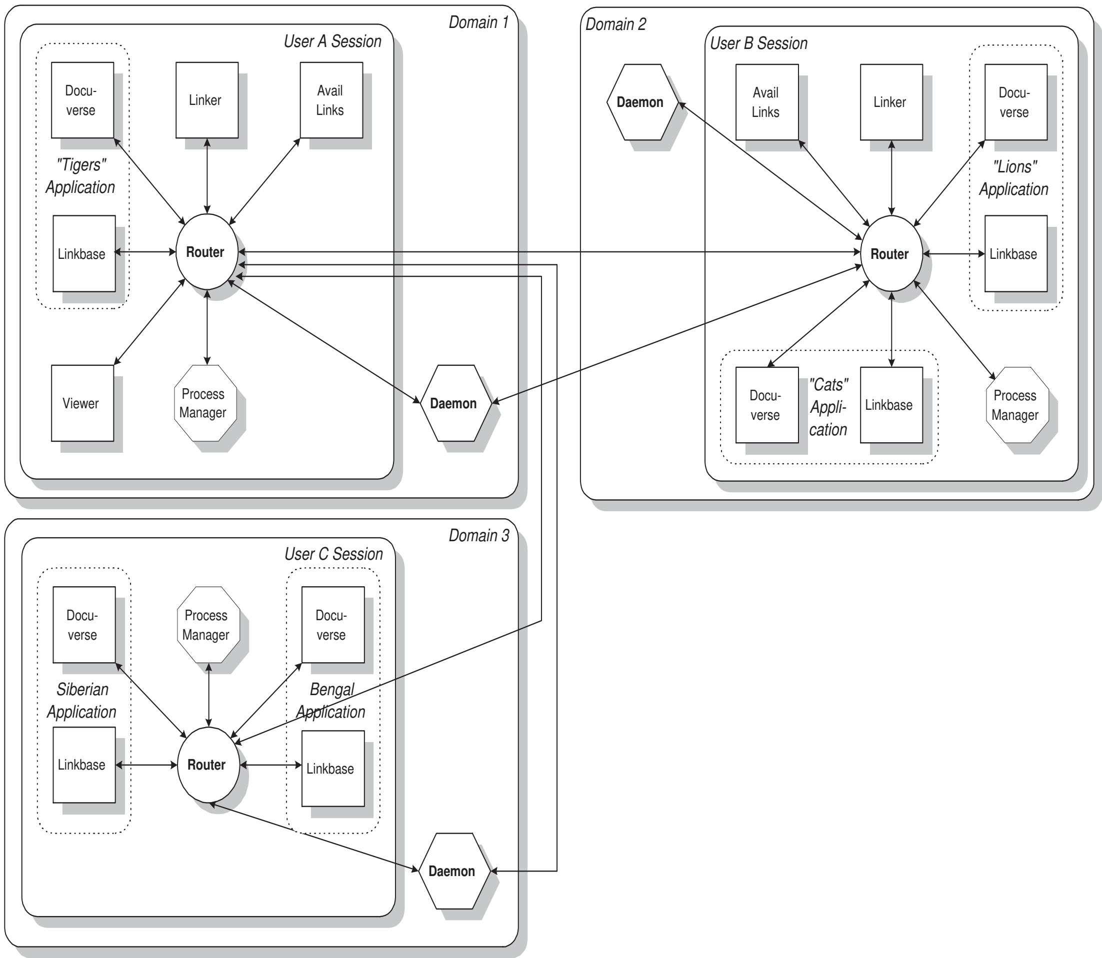


Figure 8.4: The abstraction of a hypermedia application under Microcosm TNG.

---

physical storage device they may be accessing. In a similar manner, it is possible to mount new hypermedia applications at any point within a hypermedia application hierarchy. This can be achieved either by starting new reflector processes or modifying existing ones to respecify the parent-child application relationships.

8.7 Hierarchies of Hypermedia Applications

It is well understood from the literature that hierarchies can prove useful in reducing user disorientation. Akscyn *et al* [1] made entensive use of hierarchies within KMS. Halasz [97] and Brown [29] both observed that users of NoteCards and Guide related well to hierarchical structures. Hierarchies have become recognised as a ben-

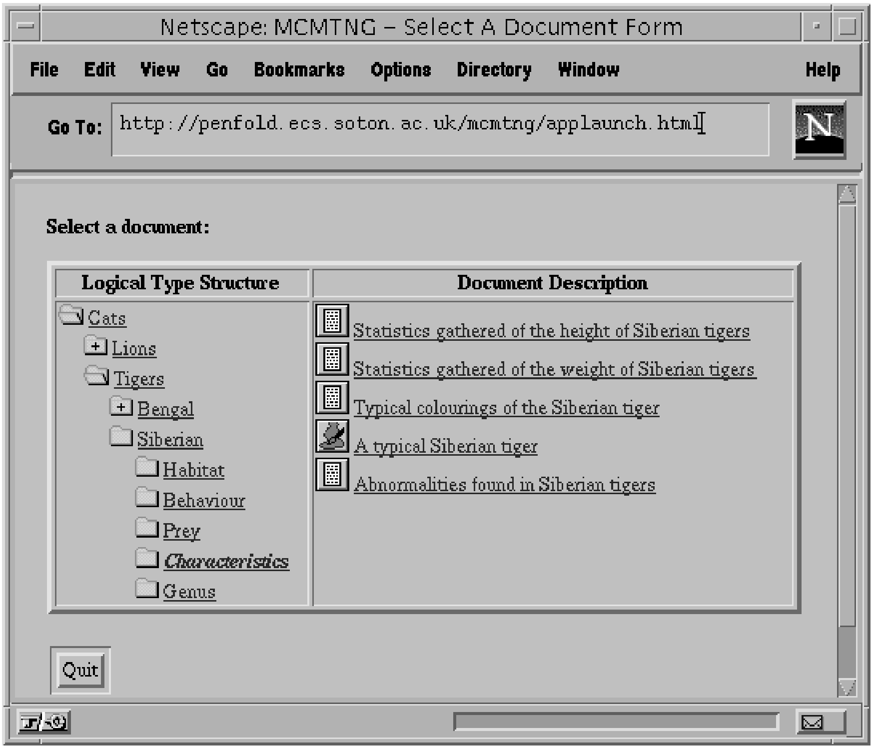


Figure 8.5: The distributed DMS browser showing a unified view of the logical document structure of the *Cats* hypermedia application hierarchy.

---

eficial adjunct to hypermedia in providing an alternative structure by which users can orientate themselves. The Microcosm TNG distributed DMS browser provides this alternative view which is of increased benefit with an expansive application or application hierarchy.

Although the *Cats* application in section 8.6.2 has been expanded to comprise a small hierarchy of distributed applications, the DMS browser can provide a unified view of the logical, but distributed, structure of the document collections. Using this interface it is possible for the user to navigate seamlessly around the aggregated document collections of the entire hypermedia application hierarchy. A screenshot of the distributed DMS browser can be seen in figure 8.5. To view a document the user simply has to click on the appropriate description in the right hand column of the table.

8.7.1 Achieving Asynchrony in the Browser

The user interface technology based upon WWW browsers described in section 7.10.3 proved an effective solution for rendering the document collection of a single docucosm process. This solution, however, could not present a unified view of the documents from multiple hypermedia applications each with multiple docucosm processes.

In order to render the document hierarchy, each of the docucosm processes that comprise the initial view are contacted to return their list of documents. An asynchronous communication model is adopted which allows the document hierarchy to be constructed and subsequently modified as a reply from each docucosm process is received. This asynchronous model *would* allow an interface to be presented to the user immediately which would be updated as each new message arrived. Unfortunately, the WWW browser used to display the user interface adheres to a strict client/server mode of interaction. This means that once an initial interface is presented there is no mechanism available for updating the display until the user interacts with the form.

To overcome the strict client/server nature of the WWW browser an alternative solution was developed using Java [171] and frames. The WWW browser is now invoked with a URL that points to an HTML page containing two frames. In one frame is a Java applet which is invisible to the user, while the other frame contains the familiar view of the distributed DMS interface. The Java applet is able to communicate with Microcosm TNG over TCP/IP sockets allowing the distributed DMS interface to be updated in an asynchronous manner.

8.8 Application Areas

Although somewhat contrived, the examples presented throughout section 8.6 illustrate the principles that underpin the approach to hypermedia application building with Microcosm TNG. In this section it is described how two large projects could benefit from the technology presented in this chapter.

8.8.1 Pirelli/FIRM

Large industrial engineering machines used by companies such as Pirelli are built from numerous components and sub-assemblies. These components and sub-assemblies are manufactured by various suppliers across the world. As any loss in production time costs money, Pirelli employ maintenance engineers to monitor and fix these large machines when they fail. The maintenance engineers rely heavily upon technical documentation to aid with the fault diagnosis and correction procedures.

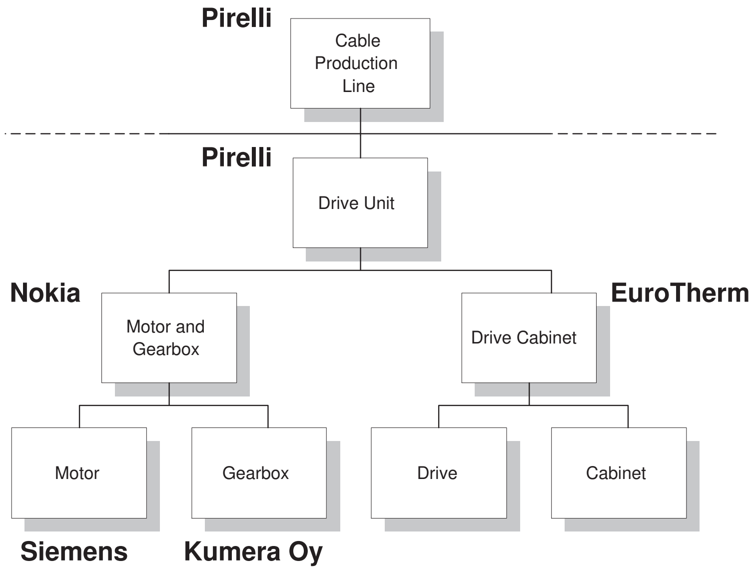


Figure 8.6: The multi-vendor hypermedia application hierarchy of a production line drive unit.

Current work on the FIRM project, introduced in section 4.11.2, includes providing hypermedia access to technical documentation for the maintenance of a cable production line at Pirelli. One machine occupying a pivotal role in this production line is the drive unit. The drive unit is built from numerous components and sub-assemblies manufactured by four different European companies. If each of these companies were to publish technical documentation on their products as hypermedia applications then the possibility would exist for a hierarchy to be devised. For example, from figure 8.6 it can be seen how the *Motor and Gearbox* made by Nokia is built with a *Motor* by Siemens and a *Gearbox* by Kumera Oy. As this relationship has been established by Nokia, any customer using the *Motor and Gearbox* is automatically afforded hypermedia access to information regarding the sub-assemblies.

The distributed nature of Microcosm TNG would allow each company to publish hypermedia applications regarding each of their products at their own site. The reflexive model allows hierarchical relationships to be forged between hypermedia applications. Components and sub-assemblies such as a motor and a gearbox are part of many larger products. As such, it is necessary that the hypermedia application associated with each component or sub-assembly can form part of other product hierarchies. The reflexive model allows hypermedia applications, such as those in figure 8.6, to participate in one or more hierarchies simultaneously and hence be used in multiple contexts.

8.8.2 SToMP

The SToMP project, introduced in section 4.11.1, was established to develop modular courseware for the teaching of physics undergraduates. The project involves nine UK universities each developing specific courseware modules in parallel.

The platform for the delivery of SToMP is the non-distributed version of Microcosm described in chapter 4. A group at the University of Surrey are the project coordinators and are responsible for directing the activities of the other groups and for integrating their contributions within a single final product. The task of assimilating the numerous contributions has not been without complications.

The SToMP application is hierarchical in structure. Syllabuses on *Waves and Vibrations*, *Measurement and Uncertainty* and *Optics* have been developed to date. Each syllabus comprises a number of blocks each of which contains several units. The structure of SToMP is sufficiently modular to allow groups to develop units in isolation. If each block or unit was built as a discrete hypermedia application, the complications involved in assimilating these distributed contributions would disappear if Microcosm TNG and the reflexive model were to be used.

8.8.3 Discussion

Using the FIRM and SToMP projects it has been discussed how both the technology and the principles that underpin Microcosm TNG and the reflexive model can help with the organisation and deployment of distributed hypermedia applications. Although this technology provides a mechanism for creating hierarchies of related information some implications of this are beyond the scope of the framework.

The technology itself does not enforce a hierarchy of hypermedia applications to adhere to a *house style* with a consistent look and feel. It might be mandatory for each division of an international company to adhere to a consistent look and feel for the corporate web site, but WWW technology also does not enforce these measures.

Inconsistent spelling and use of jargon may prove to be an issue for a hierarchy of hypermedia applications that are distributed over the UK, USA and Australia, for example. Again, this issue is also not addressed in other systems.

From a technical or pedagogical perspective it may be difficult to achieve a consistent viewpoint across a hierarchy of applications produced by multiple authors. This issue also arises when an author may consider reusing an existing hypermedia application as an alternative to writing their own. In some circumstances this may be overcome by adding supplementary material to bridge the gap. This latter approach was proven to be viable by one lecturer who reused the SToMP material on *Waves and Vibrations* for the benefit of Oceanography undergraduates.

Although it is suggested that the reflexive model can promote the wholesale reuse of hypermedia applications, it is likely that there exists a limited depth beyond which a hierarchy of hypermedia applications are able to be reused in an unmodified form. It is impossible to predict at what depth that may occur as so much is dependant upon the requirements of each particular hypermedia application.

8.9 Limitations of the Implementation

One of the attractive aspects of the reflexive model is the ability to incorporate a hypermedia application within a user session without being aware of its composition: it may be a single hypermedia application or an entire distributed hierarchy. While the advantages of this abstraction are obvious, on occasions it would be desirable to break the abstraction. For example, Microcosm TNG allows a user to target an action, such as follow link, to a specific hypermedia application. This application could be at the apex of a large hierarchy which may return vast numbers of destination documents.

As DeYoung observed [58], methods for further constraining such hierarchies to limit the search space may prove useful.

While the reflexive model provides a dynamic and versatile approach to the composition of distributed hypermedia applications, the current implementation is not infinitely scalable. If the hypermedia applications belonging to a hierarchy are closely physically located then this is not of concern, but the performance of a hierarchy that is widely distributed at each level may not prove acceptable beyond a certain depth.

A hierarchy of hypermedia applications form a Directed Acyclic Graph (DAG). The present implementation does not check for cycles in the hierarchy.

8.10 Summary

The anatomy of a hypermedia application has been examined in some depth to elicit various strategies for improving modular construction. The information can be partitioned along the dimensions of context and location in an attempt to make it more amenable to both maintenance and reuse.

It is postulated that the research community have long referred to a fictitious entity called the hypermedia application without defining it. It is then questioned as to whether the Dexter Model lacks a higher level entity to provide a logical container for hyperdocuments. Such an entity may have to address process issues in addition to data modelling. To meet this requirement the definition of a hypermedia application was established. A hypermedia application now occupies the role of a first class entity within the Microcosm TNG framework. The aim was a first attempt at providing larger, self-contained, *off-the-shelf* hypermedia components and link services.

Humans regularly employ abstraction as a mechanism for managing complexity. The abstraction of a hypermedia application allows users to construct sessions using higher level, contextual building blocks. The definition of a hypermedia application was extended to enable authors to build new applications from an arbitrary number of existing ones. This recursive definition ascribes a reflexive property to applications, hence the name of the approach.

The reflector process and the Java applet within the browser were among the minimal architectural extensions to Microcosm TNG that enabled the reflexive model to be realised. The ways in which two large research projects could benefit from the technology in this chapter have also been described.

The research described in this thesis has influenced and continues to support a number of projects at Southampton. The HCM is used with the MultiDOSm system [191] and a prototype framework for an open Graphical Information System [162]. The philosophy behind the HCM message router has been adopted as the model for communication within the commercially oriented MEMOIR system [113]. Microcosm TNG is in use by another research group as the framework for integrating an expert system with a hypermedia system to provide support for intelligent consultation in an engineering environment. Microcosm TNG has also been successfully trialed with the MAVIS project [131]. The hierarchical composition of hypermedia applications, as facilitated by the reflexive model, is a focus of continuing research within the FIRM project.

The final chapter considers the many possible extensions to the Microcosm TNG framework and the value that each may bring. A number of conclusions from the research described throughout this thesis are then presented.

9 Future Work and Conclusions

9.1 Future Work

The architecture of Microcosm was originally conceived as a vehicle for research into the issues of providing an open and flexible information environment. In this section a number of improvements to Microcosm TNG and potential areas of research are described.

9.1.1 Improving the HCM

One advantage of distributed systems is that they have the potential to be more reliable. This can be achieved through active replication. Replication can be observed in nature with living creatures often having duplicate organs (for example, two lungs and two kidneys), enabling them to survive in the event of an organ failure. In each Microcosm TNG user session a single message router is responsible for transmitting messages to the appropriate recipients. By allowing a user to start one or more message routers to serve a single session such replication can provide increased reliability and divide the processing cost.

When the HCM was under development there was very limited support for multicast across the range of platforms in the research laboratory. This situation has changed more recently, and it is also now reasonable to expect that all relatively new network routers will offer an IP multicast facility. In practice some systems have limitations on the number of groups a process can join per socket and the number of sockets a process may have open. But this technology should be experimented with as it may lead to more efficient and scalable communication.

An implementation of CORBA was evaluated during the preliminary stages of this research but it was concluded that the technology was insufficiently mature and the range of target platforms were limited. Over recent years the CORBA specification has been expanded upon significantly and a number of CORBA compliant products and services are now available. Future work in this area should include a reassessment of CORBA to determine the extent to which this *de facto* framework for distributed object technology can benefit the field of open hypermedia.

It was intended that the process manager would provide a central point at which the user could configure the system. It was envisaged that the process manager would provide an interface displaying the current processes registered with the session. If the user wished to inspect or reconfigure any process then the process manager could contact the relevant process and request an HTML form interface. The process manager could display this form and allow the user to modify the settings and return the contents of the form to the process. This approach could be used for the configuration of processes and their services in both local and remote sessions.

9.1.2 Enhancing Microcosm TNG

9.1.2.1 Adding Computer Support for Collaborative Working

The role of hypermedia within the context of Computer Supported Cooperative Work (CSCW) systems has been widely reported [169, 135, 88, 182]. In order to support the requirements of this genre of system, a number of extensions to Microcosm TNG are required.

A basic CSCW environment would provide collaborators with access to a shared whiteboard and video/audio conferencing facilities. Concurrent access to documents can be accommodated, but persistent, fine-grained locks are necessary to prevent authoring clashes [88, 182]. Mechanisms for version control are required to preserve integrity and support audit trails [94, 95]. Notification events and awareness utilities also help to keep the collaborators informed of the group dynamics.

Melly [138] describes a version of Microcosm, based on the filter architecture, that provides a shared whiteboard, versioning support and an awareness utility. With extensions to the Document Management System, the work by Melly could be combined with Microcosm TNG to provide a versatile platform for further research into CSCW.

9.1.2.2 Greater Integration with the WWW

Systems such as Hyper-G [7], DLS [39], Aquarelle [160] and Chimera [5] demonstrate the application of a distributed link service across the WWW. Recent extensions to DHM [87] offer Java-based open hypermedia authoring facilities showing a high level of integration with the WWW.

Although the peer model of Microcosm TNG is flexible, each potential user or domain requires an installation of the entire system to be present. The advent of Java has precipitated the *zero administration client*. This allows clients to be written in Java to be held at the server and downloaded to the browser on demand, thus requiring no installation.

Microcosm TNG has demonstrated integration with WWW servers through the Document Management System and gateway processes, and also with WWW clients by using them to provide user interfaces. Further work could investigate developing Java-based clients to provide every WWW user with ubiquitous access to Microcosm TNG link services.

9.1.2.3 Enhancing the User Interface

As with Microcosm, it is possible to provide proprietary document viewers for Microcosm TNG such as the primitive text viewer described in section 7.10.1. One of the primary aims of open hypermedia is to achieve a close integration with the desktop by providing access to a link service from common tools. An experiment to validate this approach was described in section 7.10.2, but in order to achieve the flexible and ubiquitous access to remote link services, as described in section 9.1.2.2, a fruitful approach would be to develop future user interfaces using a technology such as Java.

The technology by Dale [47] described in section 7.10.3 provides a simple and effective solution to the problem of distributed, heterogeneous user interfaces. Although the problem of asynchrony was overcome, described in section 8.7.1, it was recognised that a more flexible solution was required. Hemley [108] is currently developing a Java-based generic media viewer for future use with a distributed open hypermedia system. The viewer is structured such that the basic infrastructure determines the media type to be rendered and dynamically loads over the network one or more classes equipped to perform that task. This work provides an interesting direction for the provision of dynamic and flexible user interfaces for systems such as Microcosm TNG.

9.1.2.4 Support for Continuous Media

Although current technology is suitable for playing continuous media stored locally, delivering continuous media over the Internet remains a challenge as the current infrastructure of the Internet is less than ideal. In section 7.9.3.2 it has been described how the Microcosm TNG Document Management System can accommodate new protocols and delivery services as they become established.

Over recent years Internet video conferencing tools have been developed that make use of a multicast network called the MBONE. Many of these tools make use of a real time protocol (RTP) for the delivery of audio and video data. There has been much commercial activity of late with products such as Real Media from Progressive Networks, Net Show from Microsoft and StreamWorks from Xing all providing different solutions for the streaming of continuous media over the Internet. These products vary with respect to the protocols, media types and platforms supported, all factors which have a bearing on the openness of the solution.

A group led by DeRoure [55] is currently investigating the real time streaming protocol (RTSP), proposed as an IETF standard, for the streaming of audio and links over separate channels. Microcosm TNG provides a layered and modular architecture in which the integration of these technologies can be performed.

9.1.2.5 Mediated Access and Security

Microcosm TNG allows a user to specify which services of a hypermedia application are available to remote users. A more sophisticated and finer-grained access model on hypermedia services would allow a user to exercise greater control over permissions. The security model employed in Hyper-G mimics that of UNIX, allowing the owner to specify user, group and world permissions on document collections.

The Internet was originally designed for research where the emphasis was not on providing a highly secure infrastructure. Although a number of security mechanisms for the Internet have been devised [20], the next generation of the Internet protocol (IPv6) [111] will address end-to-end security as part of the standard. An orthogonal issue to security is that of payment schemes. Once security is assured, financial transactions over the Internet will become commonplace. As Nelson predicted, it is likely that future users will purchase information delivered by distributed hypermedia systems.

9.1.3 Emerging Standards

A plethora of *de facto* standards seem to be emerging that will undoubtedly have some influence upon the shape of future open hypermedia research. OLE appears to be the predominant technology for compound document architectures. The Open Document Management Alliance (ODMA) is working toward the promulgation of a standard interface to document management systems. RTSP is being promoted as the standard delivery protocol for continuous media over the Internet. It is likely that the next generation of HTTP from the WWW Consortium will also have a dramatic effect upon future developments.

Another research area that is receiving much attention at present is that of agent technology. The agent research community believe that the technology has much to offer the Internet community. DeRoure *et al* [56] are exploring the application of agent technology to distributed information management (DIM). They consider that autonomous agents can assist with resource discovery, integration, link maintenance and navigation assistance.

It is essential that future standards for open hypermedia, such as OHP [51, 4] and new reference architectures [85, 93], are inclusive of related emerging standards. Microcosm TNG can provide a suitable testbed with which to validate early specifications of new models and standards.

9.2 Conclusions

9.2.1 The Open Approach

The course of open hypermedia research is charted throughout the thesis, from the initial impetus provided by visionaries such as Van Dam, Halasz and Meyrowitz through to leading edge technology reported at the Hypertext '97 conference located in Southampton. The monolithic, proprietary closed hypermedia systems have for the most part been replaced by systems based upon modular architectures comprising loosely coupled communicating processes.

The open hypermedia philosophy states that documents are sacrosanct and that link information be held in external databases. For the architects of open hypermedia systems, the aim is for seamless integration with every application on the desktop. The goal of providing the user with an integrated information environment by providing ubiquitous access to an open link service has become more realistic over recent years.

9.2.2 Distributed Hypermedia

While certain sectors of the hypermedia community continued research into ever more flexible models and open architectures to meet diverse requirements, the World Wide Web began to attract considerable attention. The approach taken by the designers of the WWW was not in accord with current thinking among the open hypermedia community. A new document format was prescribed in which embedded links provided hypermedia navigation. Rather than seeking closer integration with the desktop tools, Berners-Lee *et al* sought closer integration with the Internet. The success in achieving the latter aim is one explanation for the unmitigated success of the WWW.

One of the key lessons to be learnt from the success of the WWW is the integration of distributed information resources. The way in which information from heterogeneous repositories can be retrieved and presented in a seamless manner has led many users to view a WWW browser as an acceptable face of the Internet. The WWW has also significantly raised the profile of hypermedia and demonstrated that it is an appropriate technology for accessing and navigating information spaces.

9.2.3 Distributed Open Hypermedia

Although the WWW can be considered an open system in some respects, there are a number of application areas for which it is unsuitable in its native form. Among the areas listed by Davis [48] are co-operative working environments, corporate information systems, technical documentation and personal information environments. To address the limitations of the WWW, a number of systems have been developed with the express intention of applying the principles of open hypermedia to the Internet, many of which are described in chapter 5.

The research reported in this thesis relates to the development of a prototype distributed open hypermedia system. Microcosm TNG retains the core philosophy of Microcosm, albeit via an alternative system architecture. The peer nature of the distributed architecture facilitates the same range of flexible configurations as described by Hill [109]. The filter chain has been replaced by a more scalable direct communication model. The new architecture of Microcosm TNG improves upon previous work by allowing a single user session to be distributed over a network of machines.

The profile of the Document Management System has been raised allowing it to become an integral component of the hypermedia system. Where previously only links could be stored in modular linkbases, documents can now also be managed in a similarly modular fashion. An interface to the Document Management System expedites the browsing of the logical document structure regardless of whether the documents reside on a local or remote host.

Integration has also been a theme throughout this research. It has been demonstrated how, arguably, the most widely used applications under Microsoft Windows and UNIX have been integrated with Microcosm TNG to facilitate cross platform linking. In addition to integration with the desktop, greater integration with the Internet has also been accomplished. FTP, Gopher and WWW were seamlessly integrated with Microcosm TNG through the Document Management System supported by an extensible array of gateway processes.

A more formal definition for a hypermedia application has been introduced to encapsulate closely related resources. A hypermedia application is defined to be a logical grouping of documents, the services that operate on the collection plus configuration details. This abstraction allows a hypermedia application to be considered a self-contained, publishable and shareable entity. It is through this entity that the Microcosm TNG architecture fulfils the three primary requirements:

distributed information sharing Hypermedia applications can be published granting access to the collection of documents therein.

distributed information organisation The logical structure of a distributed document collection of a hypermedia application can be browsed using the distributed

DMS browser.

extensible distributed services A range of supporting link services can also be supplied as part of the published application.

The anatomy of a hypermedia application has been examined in some depth to elicit various strategies for improving modular construction and judiciously grouping inter-dependencies. It has been suggested that information can be partitioned along the dimensions of context and location in an attempt to make it more amenable to both maintenance and reuse.

The formal definition of a hypermedia application allows instances to be considered as first class entities within the conceptual framework. This new meta-entity addresses one shortcoming of the Dexter Model by encompassing both the data and process facets. The definition was subsequently extended to allow one or more existing hypermedia applications to enrich the parent hypermedia application. Architectural extensions to Microcosm TNG have been described which enable the aggregation of distributed hypermedia applications and hierarchies themselves.

The research described in this thesis has influenced and continues to support a number of projects at Southampton. In addition to Microcosm TNG, the HCM is used with the MultiDOSm system [191] and a prototype framework for an open Graphical Information System [162]. The philosophy behind the HCM message router has been adopted as the model for communication within the commercially oriented MEMOIR system [113]. Microcosm TNG is in use by another research group as the framework for integrating an expert system with a hypermedia system to provide support for intelligent consultation in an engineering environment. Microcosm TNG has also been successfully trialed with the MAVIS project [131]. The hierarchical composition of hypermedia applications, as facilitated by the reflexive model, is a focus of continuing research within the FIRM project.

Bibliography

[1] R. M. Akscyn, D. L. McCracken, and Yoder E. A. A Distributed Hypermedia System for Managing Knowledge in Organisations. *Communications of the ACM*, 31(7):820–835, July 1988.

[2] B. Alberti, F. Anklesaria, P. Lindner, M. McCahill, and D. Torrey. Gopher+: Proposed Enhancements to the Gopher Protocol. ftp://boombox.micro.umn.edu/-pub/gopher/gopher_protocol/Gopher+/Gopher+.txt.

[3] B. Alberti, F. Anklesaria, P. Lindner, M. McCahill, and D. Torrey. The Internet Gopher Protocol: A Distributed Search and Retrieval Protocol. ftp://-boombox.micro.umn.edu/pub/gopher/gopher_protocol/protocol.txt.

[4] K. M. Anderson. A Critique of the Open Hypermedia Protocol. In *Proceedings of the ACM Hypertext '97 Workshop, Southampton, UK*, pages 1–4, April 1997.

[5] K. M. Anderson. Integrating Open Hypermedia Systems with the World Wide Web. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 157–166, April 1997.

[6] K. M. Anderson, R. M. Taylor, and E. J. Whitehead. Chimera: Hypertext for Heterogeneous Software Environments. In *Proceedings of the ACM Hypertext '94 Conference, Edinburgh, Scotland*, pages 94–107, September 1994.

[7] K. Andrews, F. Kappe, and H. Maurer. Serving Information to the Web with Hyper-G. *Computer Networks and ISDN Systems*, 27(6):919–926, 1995.

[8] Apple Computer Inc. Macintosh HyperCard User's Guide. Apple Computer Inc., 1987.

[9] Apple Computer Inc. OpenDoc Overview. Apple Computer Inc., 1993.

[10] Apple Computer Inc. OpenDoc Technical Summary. Apple Computer Inc., 1993.

[11] R. Aßfalg. The Proceedings of the Workshop on Open Hypertext Systems, May 1994.

[12] R. A. Bacon. The Effective Use of Computers in the Teaching of Physics. *Active Learning*, 4:37–41, 1996.

[13] R. A. Bacon and S. J. Swithenby. A Strategy for the Integration of IT-led Methods into Physics: The SToMP Approach. *Computer Education*, 26:135–141, 1996.

[14] V Balasubramanian, A. Bashian, and D. Porcher. A Large-Scale Hypermedia Application using Document Management and Web Technologies. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 134–145, April 1997.

[15] T. Berners-Lee. HTTP: A Protocol for Networked Information. http://-sunsite.doc.ic.ac.uk/rfc/rfc1945.txt, 1996.

[16] T. Berners-Lee, R. Calliau, J. Groff, and B. Pollermann. World-Wide Web: The Information Universe. *CERN*, 1992.

[17] T. Berners-Lee and D. Conolly. Hypertext Mark-up Language Specification 2.0. http://sunsite.doc.ic.ac.uk/rfc/rfc1866.txt, 1995.

[18] Berners-Lee, T. and Masinter, L. and McCahill, M. Uniform Resource Locator (URL): RFC 1738. http://sunsite.doc.ic.ac.uk/rfc/rfc1738.txt, 1994.

[19] F. Bertrand, F. Colaitis, and A. Léger. The MHEG Standard and its Relation with the Multimedia and Hypermedia Area. In *Proceedings of the IEE Conference on Image Processing and its Application*, April 1992.

[20] A. Bhimani. Securing the Commercial Internet. *Communications of the ACM*, 39(6):29–35, June 1996.

[21] A. K. Bhushan. File Transfer Protocol (FTP): RFC 0414. http://-sunsite.doc.ic.ac.uk/rfc/rfc0414.txt, 1972.

[22] M. Bieber. Issues in Modelling a "Dynamic" Hypertext Interface for Non-Hypertext Systems. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 203–218, December 1991.

[23] N. D. Bietner. Multimedia Support in Hypermedia. Master's thesis, University of Southampton, September 1993.

[24] N. D. Bietner. *Microcosm++: The Development of a Loosely Coupled Object-based Architecture for Open Hypermedia Systems*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, September 1995.

[25] N. D. Bietner and W. Hall. Putting the Media into Hypermedia. In *Proceedings of the IS and T/SPIE Conference on Multimedia and Networking, San Jose Convention Center, California, USA*, February 1995.

[26] K. P. Birman. The Process Group Approach to Reliable Distributed Computing. *Communications of the ACM*, 36(12), December 1993.

[27] A. D. Birrell and Nelson B. J. Implementing Remote Procedure Calls. *ACM Transactions on Computer Systems*, 2:39–59, February 1984.

[28] C. Bowman, U. Manber, P. B. Danzig, and M. F. Schwartz. Harvest: A Scalable, Customizable Discovery and Access System. Technical Report CU-CS 732-94, Department of Computer Science, University of Colorado, 1994.

[29] P. J. Brown. Turning Ideas into Products: The Guide System. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, November 1987.

[30] P. J. Brown. UNIX Guide: Lessons from Ten Years' Development. In D. Lucarella, M. Nanard, Nanard J., and P. Paolini, editors, *Proceedings of the ACM Hypertext '92 Conference, Milano, Italy*, pages 63–70, November 1992.

[31] P. J. Brown. Creating Educational Hyperdocuments: Can It Be Economic? *IETI*, 32(3):202–208, 1995.

[32] M. C. Buchanan and P. T. Zellweger. Specifying Temporal Behaviour in Hypermedia Documents. In D. Lucarella, M. Nanard, Nanard J., and P. Paolini,

editors, *Proceedings of the ACM Hypertext '92 Conference, Milano, Italy*, pages 262–269, November 1992.

[33] J. Buford. Evaluating HyTime: An Examination and Implementation Experience. In *Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA*, pages 105–115, March 1996.

[34] A. M. Burger, B. D. Meyer, C. P. Jung, and K. B. Long. The Virtual Notebook System. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 395–402, December 1991.

[35] V. Bush. As We May Think. *Atlantic Monthly*, pages 101–108, July 1945.

[36] B. Campbell and J. M. Goodman. HAM: A General Purpose Hypertext Abstract Machine. *Communications of the ACM*, 31(7):856–861, July 1988.

[37] S. K. Card, G. G. Robertson, and J. D. MacKinlay. The Information Visualiser: An Information Workspace. In *Proceedings of CHI '91 Human Factors in Computing Systems, New Orleans, Louisiana, USA*, pages 181–188, April 1991.

[38] L. A. Carr, H. C. Davis, and W. Hall. Experimenting with HyTime Architectural Forms for Hypertext Interchange. *Information Services and Use*, 13:111–119, 1993.

[39] L. A. Carr, D. DeRoure, W. Hall, and G. J. Hill. The Distributed Link Service: A Tool for Publishers, Authors and Readers. *Proceedings of the International World Wide Web Conference, Boston, Massachusetts, USA*, 1:647–656, 1995.

[40] K. S. Catlin, L. N. Garrett, and J. A. Launhardt. Hypermedia Templates: An Author's Tool. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 147–160, December 1991.

[41] Z. Chen, S. Tan, R. H. Campbell, and L. YongChen. Real-Time Video and Audio in the World Wide Web. In *Proceedings of the International World Wide Web Conference, Boston, Massachusetts, USA*, pages 333–348, December 1995.

[42] F. Colson and W. Hall. Multimedia Teaching with Microcosm-HiDES: Viceroy Mountbatten and the Partition of India. *History and Computing*, 3(2):89–98, 1991.

[43] J. Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, 20(9):17–40, September 1987.

[44] R. M. Crowder and W. Hall. The Use of Interactive Media as a Training and Operational Interface in the Advanced Factory. In *Proceedings of the Third International Conference on Factory 2000 Competitive Performance Through Advanced Technology, York, UK*, pages 106–110, July 1992.

[45] R. M. Crowder and P. McManus. Information Live. *IEEE Manufacturing Engineer*, 74(5):227–229, October 1995.

[46] J. Dale. The Communication Routines: A Network Layer Communication Model. Technical Report MMTR 95-2, University of Southampton, June 1995.

[47] J. Dale. Achieving User Interface Heterogeneity in a Distributed Environment. Technical Report MMTR 96-1, University of Southampton, January 1996.

[48] H. C. Davis. *Data Integrity Problems in an Open Hypermedia Link Service*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, November 1995.

[49] H. C. Davis, Hutchings G. A., and W. Hall. A Framework for Delivering Large-Scale Hypermedia Learning Material. In *Proceedings of ED-MEDIA '93 World Conference on Educational Multimedia and Hypermedia, Orlando, Florida, USA*, pages 115–122, June 1993.

[50] H. C. Davis, W. Hall, I. Heath, G. J. Hill, and R. J. Wilkins. Towards an Integrated Information Environment with Open Hypermedia Systems. In D. Lucarella, M. Nanard, Nanard J., and P. Paolini, editors, *Proceedings of the ACM Hypertext '92 Conference, Milano, Italy*, pages 181–190, November 1992.

[51] H. C. Davis, Lewis A. J., and A. Rizk. A Draft Proposal for a Standard Open Hypermedia Protocol: Revision 1.2. http://wwwcosm.ecs.soton.ac.uk/~hcd/-protweb.html, March 1996.

[52] H. C. Davis, S. J. Knight, and W. Hall. Light Hypermedia Services: A Study of Third Party Application Integration. In *Proceedings of the ACM Hypertext '94 Conference, Edinburgh, Scotland*, pages 41–50, September 1994.

[53] S. J. DeRose and D. G. Durand. *Making Hypermedia Work: A Users Guide to HyTime*. Kluwer Academic Press, 1994.

[54] D. DeRoure, L. A. Carr, W. Hall, and G. J. Hill. Enhancing Web Support for Resource Based Learning. In *Proceedings of the International World Wide Web Conference Workshop on Teaching and Training, Boston, Massachusetts, USA*, 1995.

[55] D. DeRoure, S. Goose, and N. Ridgeway. Open Hypermedia Applied to Audio. Electronics and computer science departmental research journal, University of Southampton, 1997.

[56] D. DeRoure, W. Hall, H. C. Davis, and J. Dale. Agents for Distributed Multimedia Information Management. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems in London, UK*, pages 91–102, April 1996.

[57] N. Deslisle and M. Schwartz. Neptune: A Hypertext System for CAD Applications. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 123–143, 1986.

[58] L. DeYoung. Linking Considered Harmful. In A. Rizk, N. Streitz, and J. André, editors, *Hypertext: Concepts, Systems and Applications, Proceedings of the Hypertext '90 Conference, INRIA, France*, pages 238–249, November 1990.

[59] J. J. Dongarra, S. W. Otto, M. Snir, and D. Walker. A Message Passing Standard for MPP and Workstations. *Communications of the ACM*, 39(7):84–90, July 1996.

[60] S. Duck, W. Hall, C. Pickering, and M. Riley. A Project Management Based Decision Support System for use in the Construction Industry. In *Proceedings of Information Technology Awareness in Engineering*, pages 143–154, November 1994.

[61] Eastgate Systems Inc. Storyspace: Hypertext Writing Environment for the Macintosh. Eastgate Systems Inc., 1991.

[62] D. E. Egan, J. R. Remde, T. K. Landauer, C. C. Lochbaum, and L. M. Gomez. Behavioural Evaluation and Analysis of a Hypertext Browser. In *Proceedings of CHI '89 Human Factors in Computing Systems, Austin, Texas, USA*, pages 205–210, April 1989.

[63] A. Emtage and P. Deutsch. Archie: An Electronic Directory Service for the Internet. In *Proceedings of the Winter 1992 Usenix Conference*, pages 93–110, January 1992.

[64] D. C. Englebart. A Conceptual Framework for the Augmentation of Man's Intellect. *Vistas of Information Handling*, 1, 1963.

[65] H. Fischer. OLE 2.0 Open Tools Summit. *Microsoft, Seattle W.A.*, February 1993.

[66] A. Fountain, W. Hall, I. Heath, and H. C. Davis. Microcosm: An Open Model for Hypermedia With Dynamic Linking. In A. Rizk, N. Streitz, and J. André, editors, *Hypertext: Concepts, Systems and Applications, Proceedings of the Hypertext '90 Conference, INRIA, France*, pages 298–311, November 1990.

[67] G. W. Furnas. Generalised Fisheye Views. In *Proceedings of CHI '86 Human Factors in Computing Systems, Boston, Massachusetts, USA*, pages 16–23, April 1986.

[68] R. Furuta and P. D. Stotts. The Trellis Hypertext Reference Model. *Proceedings of the NIST Hypertext Standardisation Workshop, NIST Special Publication SP500-178*, pages 83–90, 1990.

[69] F. Garzotto, L. Mainetti, and P. Paolini. Adding Multimedia Collections to the Dexter Model. In *Proceedings of the ACM Hypertext '94 Conference, Edinburgh, Scotland*, pages 70–80, September 1994.

[70] F. Garzotto, L. Mainetti, and P. Paolini. Information Reuse in Hypermedia Applications. In *Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA*, pages 93–101, April 1996.

[71] F. Garzotto, L. Mainetti, and P. Paolini. Designing Modal Hypermedia Applications. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 38–47, April 1997.

[72] F. Garzotto, P. Paolini, and D. Schwabe. HDM: A Model for the Design of Hypertext Applications. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 313–328, December 1991.

[73] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, Massachusetts, 1994.

[74] S. Goose. A Review of Selected Open Technologies. Technical Report CSTR 94-08, University of Southampton, April 1994.

[75] S. Goose. Distributed Open Hypermedia Systems. Master's thesis, University of Southampton, April 1995.

[76] S. Goose. Extending The Heterogeneous Communication Model: A Scalable and Seamless Architecture to Support Widely Distributed Process Interaction. Technical Report MMTR 95-7, University of Southampton, July 1995.

[77] S. Goose. The Design of a Generic Customisable Distributed Communication Framework. Technical Report MMTR 95-6, University of Southampton, June 1995.

[78] S. Goose and J. Dale. Layering an Open Hypermedia System Above a Distributed Communication Architecture. Poster in Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA, March 1996.

[79] S. Goose, J. Dale, W. Hall, and D. De Roure. Microcosm TNG: A Distributed Architecture to Support Reflexive Hypermedia Applications. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 226–227, April 1997.

[80] S. Goose, J. Dale, G. J. Hill, D. De Roure, and W. Hall. An Architecture to Support an Open Distributed Hypermedia System. Electronics and computer science departmental research journal, University of Southampton, 1995.

[81] S. Goose, J. Dale, G. J. Hill, D. De Roure, and W. Hall. Unifying Distributed Processing and Open Hypermedia through a Heterogeneous Communication Model. Technical Report MMTR 95-6, University of Southampton, November 1995.

[82] S. Goose, J. Dale, G. J. Hill, D. De Roure, and W. Hall. An Open Framework for Integrating Widely Distributed Hypermedia Resources. In *Proceedings of the*

IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, pages 364–371, June 1996.

[83] S. Goose and W. Hall. Enriching Audio Support in a Hypermedia System. Technical Report CSTR 94-03, University of Southampton, January 1994.

[84] S. Goose and W. Hall. The Development of a Sound Viewer for an Open Hypermedia System. *The New Review of Hypermedia and Multimedia*, 1:213–231, 1995.

[85] S. Goose, A. J. Lewis, and H. C. Davis. OHRA: Towards an Open Hypermedia Reference Architecture and a Migration Path for Existing Systems. In *Proceedings of the ACM Hypertext '97 Workshop, Southampton, UK*, pages 21–30, April 1997.

[86] K. Grønbæk. Composites in a Dexter-Based Hypermedia Framework. In *Proceedings of the ACM Hypertext '94 Conference, Edinburgh, Scotland*, pages 59–69, September 1994.

[87] K. Grønbæk, N. O. Bouvin, and L. Sloth. Designing Dexter-based Hypermedia Systems for the World Wide Web. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 146–156, April 1997.

[88] K. Grønbæk, J. A. Hem, O. L. Madsen, and L. Sloth. Designing Dexter-Based Cooperative Hypermedia Systems. In *Proceedings of the ACM Hypertext '93 Conference, Seattle, Washington, USA*, pages 25–38, November 1993.

[89] K. Grønbæk, J. A. Hem, O. L. Madsen, and L. Sloth. Cooperative Hypermedia Systems: A Dexter-Based Architecture. *Communications of the ACM*, 37(2):64–75, February 1994.

[90] K. Grønbæk and R. H. Trigg. Design Issues for a Dexter-Based Hypermedia System. In D. Lucarella, M. Nanard, Nanard J., and P. Paolini, editors, *Proceedings of the ACM Hypertext '92 Conference, Milano, Italy*, pages 191–200, November 1992.

[91] K. Grønbæk and R. H. Trigg. Design Issues for a Dexter-Based Hypermedia System. *Communications of the ACM*, 37(2):40–49, February 1994.

[92] K. Grønbæk and R. H. Trigg. Towards a Dexter-Based Model for Open Hypermedia: Unifying Embedded References and Link Objects. In *Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA*, pages 149–160, March 1996.

[93] K. Grønbæk and U. K. Wiil. Towards a Reference Architecture for Open Hypermedia. In *Proceedings of the ACM Hypertext '97 Workshop, Southampton, UK*, pages 31–38, April 1997.

[94] A. Haake. A Contextual Version Server for Hypertext Applications. In D. Lucarella, M. Nanard, Nanard J., and P. Paolini, editors, *Proceedings of the ACM Hypertext '92 Conference, Milano, Italy*, pages 43–52, November 1992.

[95] A. Haake. Under CoVer: The Implementation of a Contextual Version Server for Hypertext Applications. In *Proceedings of the ACM Hypertext '94 Conference, Edinburgh, Scotland*, pages 81–93, September 1994.

[96] B. J. Haan, P. Kahn, V. A. Riley, J. H. Coombs, and N. Meyrowitz. IRIS Hypermedia Services. *Communications of the ACM*, 35(1):36–51, January 1992.

[97] F. G. Halasz. Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, 31(7):836–852, July 1988.

[98] F. G. Halasz, T. P. Moran, and R. H. Trigg. Notecards in a Nutshell. In *Proceedings of CHI '87 Human Factors in Computing Systems, Toronto, Ontario, Canada*, pages 42–52, April 1987.

[99] F. G. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. *Proceedings of the NIST Hypertext Standardisation Workshop, NIST Special Publication SP500-178*, pages 95–133, 1990.

[100] F. G. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 31(2):30–39, February 1994.

[101] W. Hall. Ending the Tyranny of the Button. *IEEE Multimedia*, 1(1):60–68, 1994.

[102] W. Hall, H. C. Davis, and G. A. Hutchings. *Rethinking Hypermedia: The Microcosm Approach*. Kluwer Academic Press, 1996.

[103] W. Hall, G. J. Hill, and H. C. Davis. The Microcosm Link Service. In *Proceedings of the ACM Hypertext '93 Conference, Seattle, Washington, USA*, pages 256–259, November 1993.

[104] L. Hardman, D. C. A. Bulterman, and G. Van Rossum. Adding Time and Content to the Dexter Model. *Communications of the ACM*, 37(2):50–63, February 1994.

[105] L. Hardman, J. Van Ossenbruggen, and D. C. A. Bulterman. A HyTime-Compliant Interchange Format for CMIF Hypermedia Documents. in progress.

[106] I. Hayes. *Specification Case Studies*. Prentice Hall International Series in Computer Science, Prentice Hall International, 1987.

[107] I. Heath. *An Open Model for Hypermedia: Abstracting Links from Documents*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, September 1992.

[108] N. Hemley. Generic Media Viewer. Electronics and computer science project report, University of Southampton, 1997.

[109] G. J. Hill. *Extending an Open Hypermedia System to a Distributed Environment*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, June 1994.

[110] G. J. Hill and W. Hall. Extending the Microcosm Model to a Distributed Environment. In *Proceedings of the ACM Hypertext '94 Conference, Edinburgh, Scotland*, pages 32–40, September 1994.

[111] R. M. Hinden. IP Next Generation Overview. *Communications of the ACM*, 39(6):61–71, June 1996.

[112] K. Hirata, Y. Hara, N. Shibata, and F. Hirabayashi. Media-based Navigation for Hypermedia Systems. In *Proceedings of the ACM Hypertext '93 Conference, Seattle, Washington, USA*, pages 159–173, November 1993.

[113] G. A. Hutchings, G. J. Hill, R. James, S. Loades, J. Halé, and M. Hatzopulous. Exploiting Serendipity Amongst Users to Provide Support for Hypertext Nav-

igation. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 212–213, April 1997.

[114] International Standards Organisation. Hypermedia/Time-based Structuring Language (HyTime), ISO/IEC Standard 10744. International Standards Organisation, 1992.

[115] Iona Technologies Ltd. OrbixTalk Whitepaper. Iona Technologies Ltd., 1996.

[116] T. Isakowitz, E. Stohr, and P. Balasubramanian. RMM: A Methodology for Structuring Hypermedia Design. *Communications of the ACM*, 38(8):67–69, 1995.

[117] Isis Distributed Systems Inc. Object Groups: A Response to the ORB 2.0 RFI. Technical report, Isis Distributed Systems Inc., April 1993.

[118] S. Jackson and N. Yankelovich. InterMail: A Prototype Hypermedia Mail System. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 405–409, December 1991.

[119] C. J. Kacmar and J. J. Leggett. A Process-Oriented Extensible Hypertext Architecture. *ACM Transaction on Information Systems*, 9(4):399–419, 1991.

[120] B. Kahle and A. Medlar. An Information System for Corporate Users: Wide Area Information Server. Technical Report TMC-199, Thinking Machines Incorporated, 1991.

[121] P. Kahn and B. J. Haan. Video in Hypermedia: The Design of InterVideo. *Visual Resource*, VII:353–360, 1991.

[122] B. Kantor and P. Lapsley. Network News Transfer Protocol (NNTP): RFC 0977. http://sunsite.doc.ic.ac.uk/rfc/rfc0977.txt, 1986.

[123] F. Kappe, G. Pani, and F. Schnable. The Architecture of a Massively Distributed Hypermedia System. *Internet Research: Electronic Networking Applications and Policy*, 3(1):10–24, 1993.

[124] S. J. Knight. *Abstracting Anchors from Documents*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, May 1996.

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

[126] J. J. Leggett and J. L. Schnase. Dexter With Open Eyes. *Communications of the ACM*, 37(2):77–86, February 1994.

[127] J. J. Leggett, J. L. Schnase, J. B. Smith, and E. A. Fox. Final Report of the NSF Workshop on Hyperbase Systems. Technical Report TAMU-HRL 93-002, Texas A & M University, 1993.

[128] T. M. Levergood, A. C. Payne, J. Gettsy, G. W. Treese, and L. C. Stewart. AudioFile: A Network-Transparent System for Distributed Audio Applications. Technical Report CRL 93-8, University of Cambridge, June 1993.

[129] A. J. Lewis. Hypermedia Link Service as an Operating System Extension. Master's thesis, University of Southampton, April 1993.

[130] P. H. Lewis, H. C. Davis, M. R. Dobie, and W. Hall. Towards Multimedia Thesaurus Support for Media-based Navigation. In *Proceedings of the First International Workshop on Image Databases and Multi-Media Search, Amsterdam, Netherlands*, pages 83–90, August 1996.

[131] P. H. Lewis, H. C. Davis, S. R. Griffiths, W. Hall, and R. J. Wilkins. Media-based Navigation with Generic Links. In *Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA*, pages 215–223, March 1996.

[132] Z. Li. *Information Retrieval for Automatic Link Creation in Hypertext Systems*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, October 1993.

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

[134] D. Lowe, W. Hall, and A. Ginige. *Principles of Hypermedia Engineering: Beyond the Web*. To be published by Wiley and Sons, Inc., 1997.

[135] K. C. Malcolm, Poltrock S. E., and Schuler D. Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 13–25, December 1991.

[136] C. C. Marshall, F. G. Halasz, R. A. Rodgers, and W. C. Janssen. Aquanet: A Hypertext Tool to Hold Your Knowledge in Place. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 261–275, December 1991.

[137] H. Maurer and I. Tomek. Some Aspects of Hypermedia Systems and their Treatment in Hyper-G. *Wirtschaftsinformatik*, 32(2):187–196, 1990.

[138] M. Melly. *Cooperative Working in an Open Hypermedia Environment*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, September 1995.

[139] N. Meyrowitz. Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Application Framework. In *Proceedings of OOPSLA '86*, pages 186–201, September 1986.

[140] N. Meyrowitz. The Missing Link: Why We're All Doing Hypertext Wrong. *The Society of Text, MIT Press, Cambridge, Massachusetts*, pages 107–114, 1989.

[141] J. Myers. Post Office Protocol (POP3): RFC 1734. http://sunsite.doc.ic.ac.uk/-rfc/rfc1734.txt, 1994.

[142] J. Nanard and M. Nanard. Using Structured Types to Incorporate Knowledge in Hypertext. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 329–344, December 1991.

[143] J. Neilsen. *Hypertext and Hypermedia*. Academic Press, 1990.

[144] T. H. Nelson. *Literary Machines*. Mindful Press, 1987.

[145] S. R. Newcomb. Explanatory Cover Material for Section 7.2 of X3V1.8M/SD–7. *Proceedings of the NIST Hypertext Standardisation Workshop, NIST Special Publication SP500-178*, pages 179–188, 1990.

[146] S. R. Newcomb. Standard Music Description Language. *IEEE Computer*, 24(8), 1991.

[147] J. Noll and W. Scacchi. Integrating Diverse Information Repositories: A Distributed Hypertext Approach. *IEEE Computer*, 24(12), December 1991.

[148] P. J. Nürnberg, J. J. Leggett, E. R. Schneider, and J. L. Schnase. Hypermedia Operating Systems: A New Paradigm for Computing. In *Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA*, pages 194–202, March 1996.

[149] Object Management Group. Common Object Request Broker: Architecture and Specification. Technical Report OMG Technical Document 91-12-1, Revision 1.1, Object Management Group, December 1991.

[150] K. Østerbye and U. K. Wiil. The Flag Taxonomy of Open Hypermedia Systems. In *Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA*, pages 129–139, March 1996.

[151] OWL International Incorporated. Guide: Hypertext for the PC. OWL International Incorporated, 1987.

[152] M. Palaniappan, N. Yankelovich, and M. Sawtelle. Linking Active Anchors: A Stage In The Evolution Of Hypermedia. *Hypermedia*, 2(1):47–66, January 1990.

[153] H. V. D. Parunak. Don't Link Me In: Set Based Hypermedia for Taxonomic Reasoning. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 233–242, December 1991.

[154] A. Pearl. Sun's Link Service: A Protocol for Open Linking. In *Proceedings of the Hypertext '89 Conference on Hypertext, Pittsburgh, Pennsylvania, USA*, pages 137–146, November 1991.

[155] J. J. Puttress and N. M. Guimaraes. The Toolkit Approach to Hypermedia. In A. Rizk, N. Streitz, and J. André, editors, *Hypertext: Concepts, Systems and Applications, Proceedings of the Hypertext '90 Conference, INRIA, France*, pages 25–37, November 1990.

[156] P. Renaud. *An Introduction to Client/Server Systems*. Wiley and Sons, Inc., 1994.

[157] V. Riley. An Interchange Format for Hypermedia Systems: The Intermedia Model. *Proceedings of the NIST Hypertext Standardisation Workshop, NIST Special*

Publication SP500-178, pages 213–222, 1990.

[158] A. Rizk, F. Malézieux, and A. Léger. Distributed Hypermedia Link Service on a WAN: An Experiment with MHEG on the ATM Network. In *Proceedings of EuroGraphics '94 Conference on Open Hypermedia Systems, Graz, Austria*, pages 18–28, April 1994.

[159] A. Rizk and I. Sauter. Multicard: An Open Hypermedia System. In D. Lucarella, M. Nanard, Nanard J., and P. Paolini, editors, *Proceedings of the ACM Hypertext '92 Conference, Milano, Italy*, pages 4–10, November 1992.

[160] A. Rizk and D. Sutcliffe. Distributed Link Service in the Aquarelle Project. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 208–209, April 1997.

[161] G. Rossi, D. Schwabe, and A. Garrido. Design Reuse in Hypermedia Application Development. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 57–66, April 1997.

[162] J. D. Sadler. Towards New Models for GIS. Master's thesis, University of Southampton, April 1996.

[163] H. A. Schütt and N. A. Streitz. Hyperbase: A Hypermedia Engine Based on a Relational Database Management System. In A. Rizk, N. Streitz, and J. André, editors, *Hypertext: Concepts, Systems and Applications, Proceedings of the Hypertext '90 Conference, INRIA, France*, pages 95–108, November 1990.

[164] D. Schwabe, G. Rossi, and Barbosa S. D. J. Systematic Hypermedia Application Design with OOHDM. In *Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA*, pages 116–128, September 1996.

[165] D. E. Shackelford, J. B. Smith, and F. D. Smith. The Architecture and Implementation of a Distributed Graph Server. In *Proceedings of the ACM Hypertext '93 Conference, Seattle, Washington, USA*, pages 1–13, November 1993.

[166] F. M. Shipman III, R. J. Chaney, and G. A. Gorry. Distributed Hypertext for Collaborative Research: The Virtual Notebook System. In *Proceedings of the*

Hypertext '89 Conference on Hypertext, Pittsburgh, Pennsylvania, USA, pages 129–136, November 1989.

[167] B. Shneiderman. User Interface Design for the Hyperties Electronic Encylopaedia. In *Proceedings of the Hypertext '87 Workshop, University of North Carolina, USA*, pages 189–194, November 1987.

[168] J. B. Smith and F. D. Smith. ABC: A Hypermedia System for Artefact-Based Collaboration. In *Proceedings of the ACM Hypertext '91 Conference, San Antonio, Texas, USA*, pages 179–192, December 1991.

[169] N. Streitz, J. Haake, J. Hannemann, A. Lemke, W. Schuler, H. Schütt, and M. Thuring. SEPIA: A Cooperative Hypermedia Authoring Environment. In A. Rizk, N. Streitz, and J. André, editors, *Hypertext: Concepts, Systems and Applications, Proceedings of the Hypertext '90 Conference, INRIA, France*, pages 11–22, November 1990.

[170] Sun Microsystems Inc. The Hot Java Browser. http://www.javasoft.com/-products/hotjavaviews/hjv.white.html, 1995.

[171] Sun Microsystems Inc. The Java Language. http://www.javasoft.com/docs/-language_environment/, 1995.

[172] A. S. Tanenbaum. *Distributed Operating Systems*. Prentice Hall International Series in Computer Science, Prentice Hall International, 1995.

[173] C. W. Thompson. The Strawman Reference Model for Hypermedia Systems. *Proceedings of the NIST Hypertext Standardisation Workshop, NIST Special Publication SP500-178*, pages 189–196, January 1990.

[174] R. H. Trigg and K. Grønbæk. Heterogeneity, Structure and CSCW: Three Challenges for Open Hypermedia. In *Proceedings of the ACM Hypertext '97 Workshop, Southampton, UK*, pages 75–77, April 1997.

[175] R. H. Trigg, T. P. Moran, and F. G. Halasz. Tailorability of Notecards. In *Proceedings of Interact '87, Stuggart, West Germany*, pages 723–728, August 1987.

[176] URI Working Group. Uniform Resource Identifier (URI). http://-www.ics.uci.edu/pub/ietf/uri, 1997.

[177] A. Van Dam. Hypertext '87 Keynote Address. *Communications of the ACM*, 31(7):887–895, July 1988.

[178] M. J. Weal. 3D Interfaces to Hypermedia Information Spaces. Master's thesis, University of Southampton, May 1996.

[179] E. J. Whitehead. An Architectural Model for Application Integration in Open Hypermedia Environments. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 1–12, April 1997.

[180] U. K. Wiil and S. Demeyer. The Proceedings of the 2nd Workshop on Open Hypermedia Systems. Technical Report 96-10, University of California Irvine, April 1996.

[181] U. K. Wiil and J. J. Leggett. Hyperform: Using Extensibility to Develop Dynamic, Open and Distributed Hypertext Systems. In D. Lucarella, M. Nanard, Nanard J., and P. Paolini, editors, *Proceedings of the ACM Hypertext '92 Conference, Milano, Italy*, pages 251–261, November 1992.

[182] U. K. Wiil and J. J. Leggett. Concurrency Control in Collaborative Hypertext Systems. In *Proceedings of the ACM Hypertext '93 Conference, Seattle, Washington, USA*, pages 14–24, November 1993.

[183] U. K. Wiil and J. J. Leggett. The HyperDisco Approach to Open Hypermedia Systems. In *Proceedings of the ACM Hypertext '96 Conference, Washington D.C., USA*, pages 140–148, March 1996.

[184] U. K. Wiil and J. J. Leggett. Workspaces: The HyperDisco Approach to Internet Distribution. In *Proceedings of the ACM Hypertext '97 Conference, Southampton, UK*, pages 13–23, April 1997.

[185] U. K. Wiil and K. Østerbye. The Proceedings of the ECHT '94 Workshop on Open Hypermedia Systems. Technical Report R-94-2038, Aalborg University, March 1994.

[186] U. K. Wiil and E. J. Whitehead. Interoperability and Open Hypermedia Systems. In *Proceedings of the ACM Hypertext '97 Workshop, Southampton, UK*, pages 78–83, April 1997.

[187] R. J. Wilkins. *The Advisor Agent: A Model for the Dynamic Integration of Navigation Information Within an Open Hypermedia System*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, September 1994.

[188] R. J. Wilkins, I. Heath, and W. Hall. A Direct Communication Model for Process Management in an Open Hypermedia System. Technical Report CSTR 93-14, University of Southampton, 1993.

[189] N. Yankelovich, B. J. Haan, N. Meyrowitz, and S. M. Drucker. Intermedia: The Concept and the Construction of a Seamless Information Environment. *IEEE Computer*, 21(1):81–96, 1988.

[190] N. Yankelovich, N. Meyrowitz, and A. Van Dam. Reading and Writing the Electronic Book. *IEEE Computer*, 16(1):15–30, 1985.

[191] M. Zirn. MultiDOSm: Design of a Hypermedia System Based on a Distributed Semantic Network. Electronics and computer science project report, University of Southampton, 1996.