AN AGENT-BASED FRAMEWORK TO SUPPORT ADAPTIVE HYPERMEDIA

By

Christopher Paul Bailey

A thesis submitted for the degree of Doctor of Philosophy

Department of Electronics and Computer Science,

University of Southampton,

United Kingdom.

December, 2002

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING

ELECTRONICS AND COMPUTER SCIENCE DEPARTMENT

Doctor of Philosophy

AN AGENT-BASED FRAMEWORK TO SUPPORT ADAPTIVE HYPERMEDIA

by Christopher Paul Bailey

The field of adaptive hypermedia is a little over a decade old. It has a rich history in a range of fields such as artificial intelligence, user modelling, intelligent tutoring systems and hypertext. Early adaptive hypermedia work concentrated on application-led research; developing a range of systems for specific purposes. In 1996, Peter Brusilovsky reviewed the state-of-the-art and proposed a taxonomy of adaptive hypermedia techniques, thereby providing the means to categorise adaptive hypermedia systems. Since then, several practical frameworks for adaptive hypermedia applications have been produced, in addition to formal models for formalising adaptive hypermedia applications.

This thesis presents a new framework for adaptive hypermedia systems based on agent technology, a field of research largely ignored within the adaptive community. Conceptually, this framework occupies a middle ground between the formal reference models for adaptive hypermedia and application-specific frameworks. This framework provides the means to implement formal models using variety of architectural approaches.

Three novel adaptive hypermedia applications have been developed around this agent-based framework. Each system employs different architectural structures, they model the user with a variety of profiling techniques, and each provides a different set of adaptive features. The diversity of these three systems emphasises the flexibility and functionality of this proposed agent-based framework.

Contents

List of Figures

List of Tables

Acknowledgements

I'd like to thank Wendy Hall for her excellent supervision and continual faith in my abilities, Peter Boait, Gary Hall, Stuart Middleton, David Millard and Muan Ng for taking time out of their busy lives to proof-read this document. I'd also like to mention all the people I've worked with over the course of my PhD, including Samhaa El-Beltagy, David Millard, Muan Ng and Mark Weal. Special thanks go to my family, friends and all the IAM members that have helped and supported me over the last three years.

Definitions and Abbreviations Used

AH	Adaptive Hypermedia
AHS	Adaptive Hypermedia System
OH	Open Hypermedia
OHS	Open Hypermedia Systems
Web	The World Wide Web

Chapter 1

Introduction

Information is everywhere. The exponential growth of the web over a few short years has brought the wealth of human knowledge to people's fingertips. This rapid expansion has no signs of abating. Today the web is a medium for expressing ideas, stories, facts, records, interests - anything anybody ever wanted to say. The simplicity of the hypertext model behind the web is the primary reason for its success. However the sheer corpus of knowledge available to any one person is more than they can possibly absorb, causing the well documented information overload and lost in hyperspace problems.

Adaptive Hypermedia (AH) researchers produce systems and techniques aimed at reducing cognitive overload and decreasing the time users spend in finding specific information. These systems build a profile of the user, a user model, and use this knowledge to simplify the browsing experience and guide the user through an information space (called a domain) to find the right information. Drawing on advances in the fields of user modelling, artificial intelligence and hypertext research, the AH community has continued to expand and mature.

This initial chapter introduces the reader to the work documented in this thesis by presenting a brief overview of the adaptive hypermedia research field, followed by an outline of the contributions this work makes to the field. This chapter also provides a description of the thesis structure and concludes with a declaration of authenticity made by the author concerning the research documented in the thesis.

1.1 Overview of Research

Adaptive hypermedia research is over a decade old. The formative years saw the development of a host of adaptive applications and the re-implementation of adaptive features onto existing systems. In 1996, Peter Brusilovsky revolutionised the field by developing the first taxonomy of adaptive techniques. This work reviewed the existing state-of-the-art Adaptive Hypermedia Systems (AHS) and provided methods for identifying and categorising new adaptive applications. After this, the field saw an expansion with the development of a broader range of AHS, techniques and the first official conference in 2000. Today a rich range of skills propagates through the community, increasing its diversity and strengthening the research.

In addition to the applications still being developed, more and more researchers are formalising AHS and producing models and frameworks for specifying these systems. Their objective is to increase the interoperability between approaches. However, such work is drawing on old techniques developed for hypertext research from the pre-adaptive hypermedia era. For example, the AHAM model for adaptive hypertext systems (De Bra *et al.,* 1999b) is based on a hypertext model developed in 1988 (Halasz, 1994) and carries with it the limitations of the underlying model. In contrast, very little time has been spent looking at the more recent work from other fields. One example of this is the lack of agent-based adaptive hypermedia systems within the community. There are many properties of agents that make them well suited to adaptive environments.

Agents use plans or rules to govern their actions in specific circumstances while storing information and knowledge in formal knowledge databases. These features can easily be mapped onto the storage of user models, and the use of adaptive rules for personalising information to users. In addition, agents have been used in a variety of fields which have clear parallels with AHS. Personal agents, bidding agents, diary agents, mail filtering agents and recommender systems are all demonstrations of areas that could be used for adaptive hypermedia.

A second area of research outside of AH that this thesis draws upon is that of Open Hypermedia (OH). OH is a field with a similar history to Adaptive Hypermedia, both originating from the Hypertext community at the end of the 1980's. Early open

hypermedia research dealt with the separation of link structures from the contents of documents, while more recent work has been spent developing models for expressing a wide variety of hypertext systems.

The effort of researchers at Southampton University has led to the development of the Southampton Framework for Agent Research (SoFAR). SoFAR is an agent framework that promotes resource discovery, ontology-based conversation and knowledge modelling. Using this platform, an implementation-oriented framework for producing AHS has been developed as part of this thesis. The Agent-Based framework for Adaptive Hypermedia (ABAH) offers flexible solutions to developing adaptive hypermedia and adaptive web-based applications and can be used in conjunction with existing models of adaptive hypermedia. This framework conceptualizes the key components within all AHS and specifies agent roles for each of them. This, in addition to explicit knowledge models and methods of user interaction, forms a framework with which is it possible to implement any adaptive hypermedia or web-based system.

This thesis documents the production of three unique agent-based AHS, which combine agent technology with several open hypermedia techniques to provide AH. These applications demonstrate both the validity of using agent approaches in developing AHS and the benefits open hypermedia research can bring to the domain of adaptive hypermedia systems.

1.2 Contributions

This thesis documents several key contributions made to the field of adaptive hypermedia.

Primarily, the proposed agent framework is the first of its kind. It represents the first use of an agent-based framework for developing adaptive hypermedia applications. The framework also represents the first general all-purpose framework for adaptive hypermedia. While models have been developed for expressing the functionality of adaptive hypermedia applications, and several frameworks exist for implementing specific classes of adaptive hypermedia applications, ABAH lies between these approaches, providing a way to implement theoretical models without being limited to a specific type of adaptive hypermedia domain. It is also the first framework to cover

the wide variety of both hypertext and web-based adaptive systems, supporting all types of existing AHS while the agent grounding facilitates rapid system development and brings new features to the adaptive applications.

The second contribution this work makes to the adaptive hypermedia field is the introduction of two different models of context used to provide adaptive functionality. The first method uses existing machine learning and information retrieval techniques to form spatial models of the user's situation and match these up against a set of linkbases. The second approach to context originated from open hypermedia research into interoperability standards and is implemented within the contextual link server Auld Linky. This second method highlights the applicability of open hypermedia techniques bringing new perspectives, approaches and methodologies to existing AH development.

In addition to these claims, this work also contains the first example of an agent-based link service to provide adaptive hypermedia, and uses experience gained from working with open hypermedia principles to reflect on critical aspects of existing adaptive research.

1.3 Document Structure

This thesis describes the process of developing an Agent-Based Adaptive Hypermedia framework (ABAH). While early chapters document existing research and the initial work of the author in this field, later chapters are concerned with the design, evaluation and discussion of ABAH.

Chapter 2 presents the field of adaptive hypermedia and examines its history and development from the author's perspective. The various techniques of adaptive hypermedia are documented alongside a selection of example systems and, in prelude to ABAH, an examination of existing frameworks for adaptive hypermedia.

Chapter 3 presents the field of open hypermedia, a topic that arose from hypertext research at the end of the 1980's. Unlike the application led AH field, open hypermedia research looks into the issues concerning existing hypermedia systems. This work has resulted in the development of link services, produced new ways of expressing

hypermedia systems and examined methods of increasing interoperability between hypertext systems.

Chapter 4 describes the diverse field of software agents; another influencing factor in this thesis. Originating from work in artificial intelligence, agent-based computing extends existing object-oriented paradigms to present new approaches to developing distributed systems. Agent-based computing attracts many diverse opinions, this chapter describes some of the them, including the problems and benefits associated with agent-based research.

Chapter 5 presents an overview of recommender systems; an independent research field that incorporates many of the techniques found within adaptive hypermedia, but differs in its heavy reliance on agent technology. These first 4 chapters complete the overview and background research sections of this thesis.

Chapter 6 documents the author's early work in building AHS using agents. These systems incorporate principles from the open hypermedia field to power adaptive hypermedia applications. PAADS and ALIC are two agent-based systems developed by the author which use various techniques to capture and maintain user models and provide link augmentation by matching user profiles to link databases. Integral to both systems is the concept of link services; an open hypermedia approach to creating and maintaining independent sets of links.

Chapter 7 details the development of ABAH, an agent-based framework for adaptive hypermedia applications. The early sections state the need for a new framework and identify the core components of existing AHS. Agent roles are then assigned to these components before being combined to form the ABAH framework. This chapter concludes with an analysis of some of the advantages that this framework provides, such as a new definition for adaptive web-based systems, a flexible approach to system development and an open architecture for future agent-based AHS.

Chapter 8 evaluates ABAH with the development of HA^3L, an adaptive web site that incorporates an open hypermedia contextual link server into the ABAH framework to provide several forms of adaptation. This is followed by a discussion of the role

ABAH would take in implementing systems expressed in the formal adaptive hypermedia model AHAM, and how it could be used to realise an existing classical adaptive hypermedia application such as INTERBOOK. The chapter finishes with a critical look at situations where a framework such as ABAH would be unnecessary or impede system development.

Chapter 9 concludes the thesis, summarising this work and reflecting on ABAH, its advantages, disadvantages and implications for future AH system designers. This chapter also presents some of the possible areas of future work, in terms of extending ABAH in the short-term, and a look towards future for the World Wide Web and the role agents and adaptive hypermedia will have in its development.

1.4 Declaration

This thesis describes the research undertaken by the author while working within a collaborative research environment. This report documents the original work of the author except in the following sections. Section 6.4 incorporates the TFIDF algorithm developed by Samhaa El-Beltagy for the QuIC project (supported by EPSRC grant GR/M77086). Section 8.2 describes the involvement of Auld Linky, a contextual link server produced in conjunction with the EQUATOR project by the researchers David Millard, Danius Michaelides and Mark Weal (supported by the EPSRC grant GR/N15986/01). Section 8.2.5 presents a discussion on the reflections of using FOHM to support adaptive techniques. The discussion was assisted by David Millard and Mark Weal.

Chapter 2

Adaptive Hypermedia Background

2.1 Introduction

The following three chapters present a background to the various research fields which have influenced this work. This first chapter centres on the field of Adaptive Hypermedia (AH) which is the primary research area of the work documented in this thesis. This chapter introduces the field and details some of the techniques and developments that have been made since its creation. As this work focuses on describing a framework for adaptive hypermedia, the second half of this chapter looks at the existing application-based and theoretical frameworks that have been developed for AH.

2.2 Hypermedia

In 1945 Vannevar Bush, a science advisor to US president Roosevelt, wrote a pioneering paper titled 'As We May Think' (Bush, 1945) in which he describes an automated library of records indexed not sequentially but associatively thereby mimicking the structure of the human brain. This machine, which he called the *Memex*, would be designed to augment human memory, providing its user with instant access to books, film, photographs, newspapers and whatever else would interest them. As they interact with the machine, people could build up *trails* through this information space, which could be annotated and shared with others to incorporate into their own Memex records. Vannevar's machines would form huge repositories of human knowledge and be accessible to all; from lawyers and patent attorneys to physicians and historians.

The Memex, while futuristic and fanciful in 1945, provided researchers with a completely new design approach for navigating large-scale electronic documents. The

1960s, saw the development of the first hypertext oNLine System (NLS) (Engelbart, 1963), see Figure 2-1, although it wasn't until 1965 that the term *hypertext* was coined by Ted Nelson (Nelson, 1965).

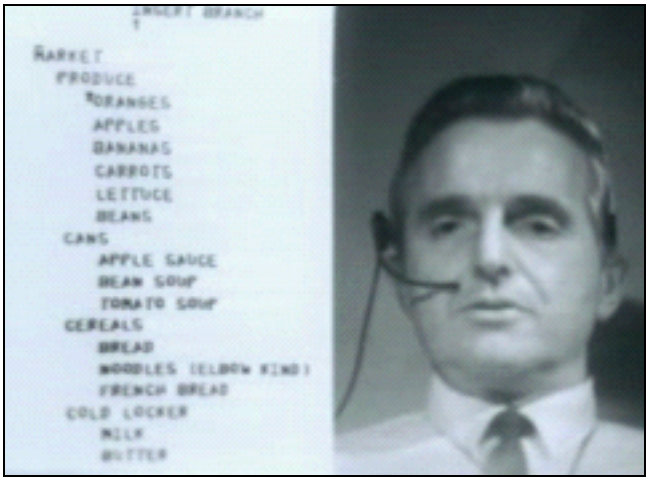

Figure 2-1. Doug Engelbart demonstrating the NLS in 1968

In the following years, many different hypertext systems were developed and as different media became cheaper and more accessible, hypertext systems evolved into *hypermedia* systems (a term also invented by Ted Nelson). Today the terms hypertext and hypermedia are used interchangeably. In 1994 Tim Berners-Lee's concept of a simple hypertext client-server approach took off on a global scale (Berners-Lee, 1999) producing what today is known as the World Wide Web (WWW) or more simply the Web. The Web, like any hypermedia system, uses the concept of *documents*, *nodes* or *pages* interrelated by a set of navigational hyperlinks (or simply called links). When reading a document, links are presented to the user who can choose to follow any one and be re-directed to a new document containing related information. This associative relation between information is an essential component of all hypermedia systems (Lowe & Hall, 1999). Grouping these traversed links in chronological order forms a *user trail*, identical to those envisaged by Vannevar Bush half a century before the Web.

In the decade since its birth, the Web's popularity has continued to grow and today there are an estimated 544.2 millions users online (NUA, 2002) and more than 550×10^9 publicly accessible documents (Bergman, 2000). However this incredible

popularity also highlights the shortcomings in the Web, issues which hypertext researchers identified in the late 1980's.

- *Information Overload*. Large hypertext systems present users with more information then they can reasonably absorb in any one session. This situation, which is becoming an increasing problem on the Web, is often referred to as information overload.

- *Lost in Hyperspace*. The second difficulty facing hypermedia systems is that their navigational structures are often poorly conceived or built in an ad-hoc manor, causing users to get mentally disorientated while they browse through many documents. This is known as the "lost in hyperspace problem" (Conklin, 1987)(Nielsen & Lyngbaek, 1990).

To tackle these problems, hypermedia systems need to guide users through the information by bringing to the user's attention documents that contain relevant, useful or interesting data. This task requires a substantial amount of understanding about the user, not just discovering their goals and interests, but also in determining how much help to give the user. Future hypermedia systems will need to build upon Bush's vision by understanding the needs of their users and providing the intelligence needed to deliver this information appropriately.

2.3 Adaptive Hypermedia

While many definitions exist for the term adaptive hypermedia, there is little disagreement within the community about what AH means. In his recent review of the field, Brusilovsky states the following.

> *"Adaptive hypermedia systems build a model of the goals, preferences and knowledge of each individual user, and use this model throughout the interaction with the user, in order to adapt to the needs of that user."* (Brusilovsky, 2001)

Whereas traditional ‘static’ hypermedia systems show the same content and links to all users, AH systems maintain a repository of knowledge about their users, known as a user model, and use this information to adapt the presentation of the hypermedia system. This adaptation can change or personalise the content of a document; the information provided to the user, or guide the user through the information by providing links to other documents or information that may be of interest to the user; tailoring the navigation between documents. Adaptive hypermedia systems tackle both the information overload and lost-in-hyperspace problems described above. Recent technological advances in the miniaturisation and mobilisation of computing devices and increasing use of web standards has seen the field of adaptive hypermedia expand to include a wider range of applications which this thesis will term adaptive web-based systems. Although the devices and applications might change, adaptive web-based systems sill rely on exactly the same principals as developed for traditional AH.

2.3.1 The Origins of Adaptive Hypermedia

1950-1980

The history of AH begins in the AI field of the late 1950’s and early 1960’s. Key players such as Alan Turing: the Turing machine (Turing, 1950), Marvin Minsky: Neural Networks, Symbolic systems and the Society of Mind, John McCarthy: Logical AI (McCarthy, 1959) and Allen Newell: the General Problem Solver (Newell *et al.*, 1960), led AI research with the belief that it would soon be possible to make computers ‘think’ as humans did. The work in programmed instruction and teaching machines led to the first Computer Assisted Instruction (CAI) systems that generated sets of problems designed to enhance student performance (Urban-Lurain, 1996)(Venezky & Osin, 1991). It was during the 1970’s, that CAI systems were empowered with the first student models designed to help the system anticipate the student’s responses. While the realisation of the limitations of existing computerised systems brought about the decline of AI in the 1970’s and early 1980’s, CAI research continued to grow with input from educational and psychology experts.

1980-1990

In 1982, the book ‘Intelligent Tutoring Systems’ (Sleeman & Brown, 1982) reviewed the state-of-the-art in CAI systems and first coined the term Intelligent Tutoring Systems (ITS). ITS were defined as those systems that monitor, coach and

instruct students. ITS became test beds for AI researchers seeking to refine their representation of the learner. However, it quickly became clear that these models also needed to include information about the student's knowledge of the information domain. ITS research continued to develop throughout the 1980's and features such as computerised assessment were introduced as a means of promoting more effective learning (Hammond & Allinson, 1989). These electronic tests gave detailed feedback to the user model and allowed the ITS to provide the notion of concept re-visitation.

1990-2000

When the Web became the primary hypermedia system for information delivery in the early 1990's, the next generation of ITS's, called adaptive educational hypermedia systems, took advantage of this new format which provided the best solution to organizing on-line learning material (Brusilovsky, 2000). This work, together with the user modelling techniques, led to Adaptive Hypermedia Systems (AHS) that combined ideas from each of these fields. AHS refer to any system that captures the features or actions of a user and uses this information to provide some form of visible information adaptation (Brusilovsky, 1996b). The focus of these systems moved away from student education and now covers a much wider range of topics such as information systems, electronic archives, online help systems, personalised news and TV guide. While there is much overlap in the techniques used in both ITS and AH systems, each is still considered to be an independent research field (Murray, 1999).

2.4 Techniques of Adaptive Hypermedia

In his seminal paper of 1996, Brusilovsky (Brusilovsky, 1996b) developed a taxonomy to help identify and classify AHS. He also reviewed the current state of the art and helped define a set of criteria for AH. The taxonomy provided a way to segment adaptive research into two categories, which Brusilovsky labelled *Adaptive Presentation* and *Adaptive Navigation Support*. Adaptive presentation, the first technique developed for AH, alters the content of the document for different users, while adaptive navigation support tailors the links between these documents.

The terms and definitions used in the taxonomy have been under some debate within the community (De Bra, 1999)(De Bra *et al.*, 1999a)(Bailey *et al.*, 2001)(Bailey

et al., 2002) however it quickly became the defacto standard for describing AH technologies due in part to a lack of alternatives. An update to the taxonmony in 2001, shown in Figure 2-2, widened the taxonomy in light of more recent research (Brusilovsky, 2001), although it retained its syntactic problems. Additional issues with this taxonomy are discussed in Section 8.2.5

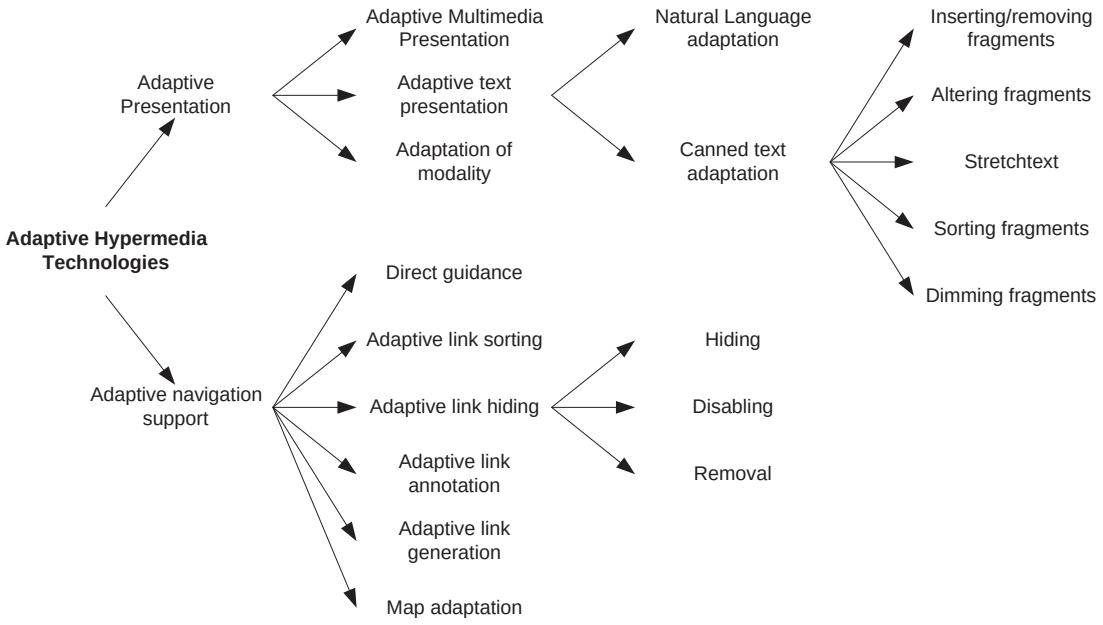

Figure 2-2. The updated taxonomy of adaptation hypermedia technologies (Brusilovsky, 2001)

2.4.1 Adaptive Presentation

Adaptive presentation is a collection of techniques for altering the content of information to the needs of a particular user, or group of users. Most adaptive presentation systems employ the idea of a concept. A concept is a chunk of information or atomic unit (De Bra, 1999) that describes an item of knowledge. System designers usually decide the level of granularity of these concepts, and the most common types include pages, paragraphs, sentences, images, audio clips or even whole documents.

The major problem with adaptive presentation systems is that they require a very high level of domain understanding if they are required to change the presentation of concepts. This prior knowledge usually consists of explicit relationships between concepts. Examples of these relationships include: A is a pre-requisite of B, A is a type of B, A is an example of B and A&B describe the same concept. For example, a page describing the theory behind ‘nuclear reactions’ will require that the user first

understands ‘basic chemical interactions’ (pre-requisite relation) and might provide links to related information such as ‘nuclear fusion’ (type-of relation), and ‘nuclear fission’ (type-of relation).

Due to the need for a large amount of prior knowledge about the information domain, systems employing adaptive presentation techniques are usually restricted to providing tailored content for an individual web site. In an attempt to reduce this problem, adaptive presentation systems try to achieve domain independence whenever possible using content abstraction.

Canned Text Adaptation

Although there are various techniques under the banner of adaptive presentation, the majority of work in this area has been categorised by Brusilovsky as canned text adaptation. This term includes fragment processing; such as fragment altering, inclusion, removal, sorting and dimming. Stretchtext, also to be found in this section, is a technique where fragments are embedded in a web page and initially hidden to the user, however the system will show these fragments inline at the request of the user.

Systems that provide adaptive presentation include AHA! (De Bra & Calvi, 1998), which can insert, remove, alter and dim fragments, HYPADAPTER (Hohl et al., 1996), which alters, inserts and removes fragments, and the SaD system devised by Hothi (Hothi, 2001) which implemented a method of fragment shading, later termed fragment dimming by Brusilovsky (Brusilovsky, 2001).

2.4.2 Adaptive Navigation Support

Adaptive navigation support modifies or augments the existing set of hyperlinks shown to the user to aid them in finding relevant information. While adaptive presentation changes the informational content of a document, navigation support changes the structure of the relationships between documents. This branch of the adaptive taxonomy contains several different techniques which can be used individually or combined to provide navigational support.

Link Sorting

Hyperlinks are sorted with the topmost items having the most relevance to users. For each user, the system assigns a relevance weight to every link based on the contents of the user model, and this weighing forms the basis of a sorting algorithm. Reordering links has the potential to damage the existing page's navigational structure, and as a result the AHS that implement link sorting, such as ELM-ART II (Weber & Specht, 1997), INTERBOOK (Brusilovsky *et al.*, 1998) and HYPADAPTER (Hohl et al., 1996), create their own set of dynamic links which can then be safely reordered with no adverse affects to the original page.

Link Annotation, Link Hiding and Direct Guidance

Link annotation is a very powerful technique that changes a link's text, style, and appearance. If the link's destination is determined to be undesirable for the user, for instance if the destination page uses terminology that a user doesn't understand, then links can be rendered in the same style as the surrounding text, effectively hiding the link even though it is still active (link hiding). Emphasizing a link with a brighter colour, a bigger font or with an additional icon can indicate a link's importance or usefulness to the user and act as a form of direct guidance. Like link sorting, care needs to be taken when annotating links so as not to upset the original style of the document. An example of link annotation can be seen in the INTERBOOK screenshot in Figure 2-3.

Link annotation is a popular technique and can be found in many systems including AHA! (De Bra & Calvi, 1998), INTERBOOK (Brusilovsky *et al.*, 1998), ELM-ART II (Weber & Specht, 1997), CHEOPS (Ferrandio *et al.*, 1997), COOL Links (Wantz & Miller, 1997) and Personal WebWatcher (Mladenic, 1996).

Link Augmentation

The technique of link augmentation involves the insertion of additional links (to useful or related information) to an existing page. These links are added on top of the existing hyperlinks found in the source document. With link augmentation, users see the page's original hyperlinks but are also presented with a set of additional links, which are often embedded in the text of the page. While link augmentation is not to be found in Brusilovsky's list of adaptive navigation support methods (Figure 2-2), it is a

technique that has been used in several AH systems (Maglio & Farrell, 2000)(Bailey & Hall, 2000)(Bailey et al., 2001)(Bailey et al., 2002). In comparison to link sorting or annotation, link augmentation is less destructive to the original format of the page because the page's existing navigational hyperlinks remain unaltered. The main issue with link augmentation is in making sure that the user does not face a navigational overload, as the page could appear to have a greater number of links on it.

2.5 User Modelling

User Modelling (UM) is the process of building up knowledge about a user in an effort to form a representation of their mental state. Typical information stored in a UM includes a person's knowledge, expertise, skills, desires and goals. The field of UM provides techniques to collect user information, maintain user profiles and deploy this information to provide personalised services to users. It is applicable in a wide range of disciplines including among others, Adaptive Hypermedia, Recommender systems, Intelligent Interfaces, Human-Computer Interactions and Expert systems (Kobsa, 1993).

To populate user models, systems can employ either explicit or implicit modelling techniques. When using an explicit approach, data is obtained by directly questioning users or requesting ratings about various products or services. While this method obtains accurate results (Watson & Sasse, 1998), it can disrupt normal browsing behaviour (Claypool *et al.,* 2001) and reduce the quality and quantity of feedback if there is a lack of perceived benefit to the user (Grundin, 1994).

In contrast to this, the non-intrusive method of implicit modelling relies solely on the observations of the user's actions. Systems track users and record their activities as they interact with the system. Activities such as reading a document, printing a document, searching a domain or purchasing a product can be used as indications of a users' desires. Such over-the-shoulder observations, while never as accurate as explicit questioning, can always be combined with explicit techniques to produce a more complete user model (Claypool *et al.,* 2001).

Once created, it is possible to process user models using a variety of existing machine learning techniques. If a system is providing collaboration-based

recommendation, it is useful to group or cluster user models together using a classification algorithm such as a neural network, k-nearest neighbour clustering, or Bayesian classifier. Alternately if a model contains a user's trail, analysis of this trail using an information retrieval techniques such as Latent Semantic Indexing (LSI) (Deerwester *et al.*, 1990), or Term-Frequency / Inverse Document Frequency (TFIDF) (Salton, 1991) analysis may yield further information about the user interests (Mladenic, 1999).

2.6 Examples of Adaptive Hypermedia Systems

The primarily application-led adaptive hypermedia research has produced many different adaptive hypermedia systems, although the majority of them are academic research efforts based on the variety of existing web formats, protocols and technologies. With the advent of mobile phones and personal digital assistants, which have a restricted screen size, limited memory and reduced bandwidth, new applications and techniques are being produced with these devices in mind (Billsus *et al.*, 2002)(Cotter & Smyth, 2000). As an exhaustive review of AH is out of the scope of this thesis, three classic systems have been chosen to demonstrate how many of the adaptive techniques from Brusilovsky's taxonomy have been implemented in existing hypermedia systems.

2.6.1 INTERBOOK

One of the most successful systems employing adaptive navigational support is INTERBOOK (Brusilovsky *et al.*, 1996b)(Brusilovsky *et al.*, 1998)(Wu, 2002), which provides the means for authoring and delivering electronic textbooks on the web. INTERBOOK resides on a web server and provides pages to any frame-enabled browser. INTERBOOK stores a domain model of concepts and their structure, and a specific type of student model called an overlay model. Overlay models allow the system to independently measure the user's knowledge in different topics. This model is built from the pages a user visits. The authors of an INTERBOOK application create the domain knowledge, organised hierarchically, along with concept information, and requirement and outcome relationships between the domain and concept information. This knowledge can then be used by INTERBOOK to generate pages, tailor font styles and place different coloured icons next to each page to represent their status to the user

based on their user model (Figure 2-3). Pages that offer no new information are marked as nothing-new (white), pages with the pre-requisite condition met are labelled as 'ready-to-be-learned' (green) and pages that fail any pre-requisite conditions are labelled as 'not-ready-to-be-learned' (red). Visited pages have a check mark placed next to them. This allows the user to visually gauge a link's condition and then make a more informed choice as to which page to visit next.

In addition to this adaptation, INTERBOOK also provides several other features to the user. The system provides direct guidance in the form of a 'Teach me' button by applying a set of heuristics to automatically choose the most suitable node from those ready to be learned. There is a 'Help' button which provides adaptively-sorted links to sections that present information about background concepts of the current section. INTERBOOK also supports a glossary index of concepts. Links to glossary items are generated automatically by INTERBOOK, and each glossary item contains a set of links back to pages describing that concept.

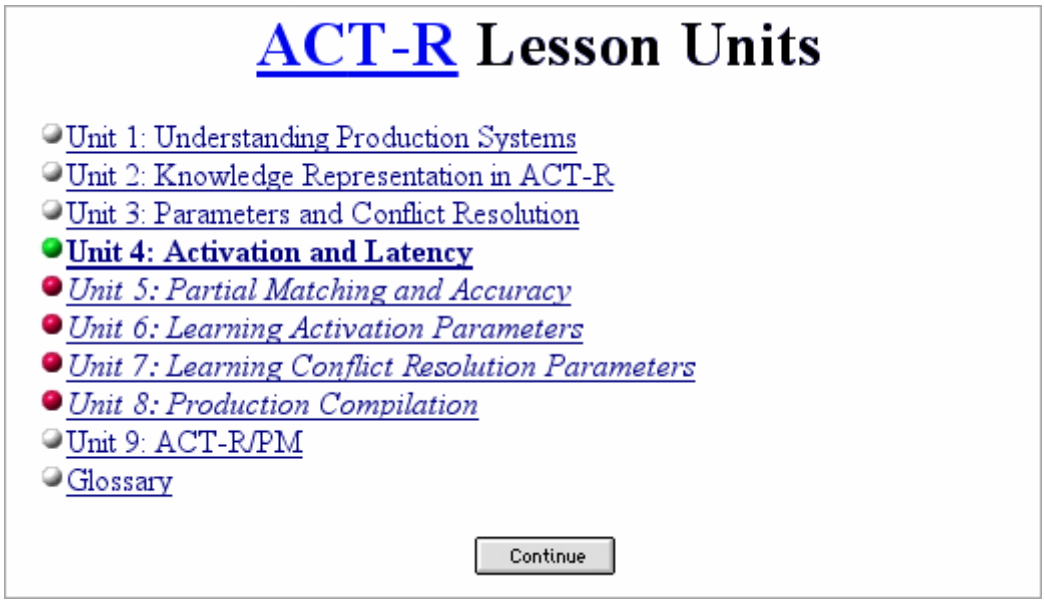

Figure 2-3. An example of adaptive link annotation in INTERBOOK

INTERBOOK is implemented using a set of CGI scripts and provides authoring tools in the form of Microsoft Word templates, which allow authors to convert their existing documents into a form that INTERBOOK can adapt.

A major study of INTERBOOK undertaken in 1998 (Brusilovsky & Eklund, 1998), found that the participants who put their faith in the system to decide the best route, followed less linear paths. They also spent longer on pages marked as 'ready-to-be-learned' and skipped quickly though those labelled as 'nothing new'. Although these results look promising, there were two major inhibiting factors in the experiment. The first was that only 17 people spent long enough on the system to warrant using their data. The second problem is that the adaptive features of INTERBOOK contributed only 20% of the total navigational tools on offer to the students, which led students to use more non-adaptive features than desired. This became obvious when analysing the non-adapted Continue button, which has a higher usage rating than all the other navigational tools combined. While not the only study of INTERBOOK, this is one of the most comprehensive and its conclusions continue to be relevant for future AH system designers.

2.6.2 AHA!

The Adaptive Hypermedia Architecture (AHA!) (De Bra & Calvi, 1998) developed in 1998 and which has since undergone several revisions (De Bra *et al.*, 2000)(Wu, 2002), is a server-side architecture for creating adaptive hypermedia applications on the web. To create an AHA! application, authors write a set of web pages which also contain conditional blocks of XHTML code, while at the same time creating 'concept' documents for each page to specify the required user knowledge for that page. The concept pages also contain the processing actions to apply to the user model once the page has been read. The AHA! system internally stores a user model which results from the pages concepts the user has read. The adaptive content on new pages are provided by combining the concept rules, conditional statements and user data together. Since its initial development, the experiences with AHA! led the authors to form an abstract model for AH systems, AHAM (see Section 2.7.1), and recent work on AHA! has seen it more closely adhere to this framework (De Bra *et al.*, 2002).

The conditional statements are a powerful way of producing a range of adaptive techniques. Each statement contains valid XHTML data which includes links, images paragraphs or entire pages of data. Each statement specifies which concepts are needed to display the data inside. An example of a conditional statement is shown below.

```
<if expr="(aAHS.introduction=80) && ((aAHS.history>50) || (aAHS.history=50))">
    <block>
        This text will only appear if the user has gained the required knowledge
        in the 'introduction' and 'history' concepts.
    </block>
    <block>
        If this is not the case then this alternative is presented instead.
    </block>
</if>
```

These statements were designed as a means to provide adaptive presentation, although adaptive link annotation and link augmentation can also be implemented as any XHTML can be placed inside the conditional text. Single words, paragraphs, hyperlinks, images and movies can be constrained to the concepts understood by the user.

When all the pages have been written, the author then writes a series of concept documents that formalise the knowledge within the system. These documents can specify pre-requisite information for a page, and any post-access actions, or they can be used to store code that is applied to a set of pages. This information is then used by AHA! to create the adaptation. Users access the web pages through a set of java servlets. These servlets apply the rules in the concept documents to the current user model. If the user is allowed to view the page, AHA! then applies the user model to the conditional rules in the accessed page to produce a personalised version of that page. The user model is built from post-access rules also specified in the concept documents. This knowledge is stored as concept-value pairs with integer values between 1-100.

AHA! also lets the author specify link classes, which are rendered in different colours with cascading style sheets. This allows an additional form of adaptive link annotation and makes it easy to provide link hiding. This hides complex information from the user while still providing access to that information if required.

2.6.3 PUSH

PUSH (Espinoza & Höök, 1996) is a web interface, via a Web browser, to an on-line manual containing several thousand individual web pages. It uses a number of AH

techniques to provide the user with a clear view of the domain and help reduce information overload.

PUSH relies on the user to provide most of the adaptability of the system. It generates two ways to navigate through the information space; menus and graphs, although neither method relies on traditional hyperlinks. Menus are defined by the system prior to run time, and provide a reliable way of letting the user pose questions to the system such as 'Describe process *X*' or 'Provide an example of *Y*'. Questions appear as a drop down list menu list that the user can select. If the user wants to submit a question that is not on the list, then they may enter it into a text box provided by the system.

The second technique is a form of map adaptation. When the browser loads a new page, the PUSH system processes the local domain structure and constructs a graph displaying all objects related to the current page as well as their relative positions. This provides a very clear view to the user of their current position and gives an idea of the local information space around them. By making each node of the graph a pointer to a web page describing it in more detail, the graphs become the primary method of navigation within the information domain.

Each page, which has been generated dynamically with Common Gateway Interface (CGI) scripts, contains a textual description of a subject. This information is specific to the goal of the user and is obtained, either by deduction or prior expression by the user (Brusilovsky, 1996b). PUSH then augments this with a list of hyperlinks describing other aspects of the concept. When the user clicks on one of these, instead of navigating to a new page, the system inserts the new information just below the current text. By clicking on the hyperlink again, the description is removed. This stretchtext technique essentially allows the user to create their own view of the web pages, adding or removing information and thereby providing the adaptation themselves. The idea is extended further into what Espinoza and Höök call 'hotwords'. These are words in bold that, when clicked on, yield questions relating to the word that the user might want to ask. When a question is selected, the answer appears in the browser window below the current information. This additional information can aid the user in understanding the concept.

2.7 Models and Frameworks for Adaptive Hypermedia

For over a decade the AH community has produced a wide range of applications in many varied domains, however there are very few frameworks for AH and even fewer models. The models describe, at a conceptual level, the components and descriptive languages used for specifying generalised adaptive hypermedia systems. The frameworks represent real systems that the authors claim can be used in a variety of adaptive applications, however in practice they are usually limited to a specific application or area, such as adaptive learning environments, adaptive hyperbooks or distance learning courses.

2.7.1 AHAM

The Adaptive Hypermedia Application Model, AHAM (De Bra *et al.*, 1999b) was developed after work on AHA! (see Section 2.6.2) highlighted the desire to produce a reference model that could be used to express the functionality of any adaptive hypermedia system (De Bra *et al.*, 2002). Since the development of AHAM, AHA! has been further modified to better fit into the AHAM model.

AHAM is based on DEXTER, a formal reference model for hypertext systems developed in 1988 (Halasz & Schwartz, 1994). The goal of DEXTER was to produce a model that captures the important formal and informal abstractions found in existing hypertext systems, as well as provide a common set of terms for describing such systems. The model allows the properties of a system to be expressed in DEXTER and then compared against the functionality of other systems. It was the hope of the designers for this model to be the basis for interoperability standards for hypertext systems. DEXTER separates the components of a hypertext system into three major layers; the *within component layer* which stores the contents of the domain, the *storage layer* which contains the structure (nodes and links) between objects in the component layer, and the *runtime layer* which presents the hypertext information to the user. The DEXTER model also includes two smaller layers; an *anchoring* layer to allow addressing of individual chunks of data within the component layer, and a *presentation specification* layer to provide the runtime layer with information on how to present specific hypertext components.

While DEXTER would successfully describe most hypertexts of the time, indeed the web today can be mapped onto this model, the authors of AHAM were driven by the realisation that DEXTER needs to be extended if it is to be used to describe the new field of AH. The AHAM structure uses identical within component, anchoring and runtime layers, however the AHAM model modifies the storage layer to include a user model, a domain model and an adaptation model as seen in Figure 2-4.

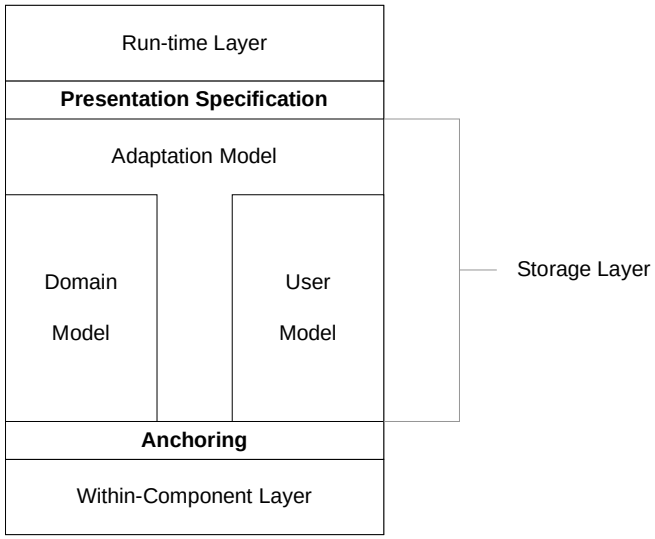

Figure 2-4. The AHAM Model (Wu *et al.*, 2001)

The domain model contains atomic concepts, which describe components within the component layer, and composite concepts, which contain a set of atomic concepts. In AHAM, these concepts can represent whole pages or fragments of pages. All concepts are uniquely addressed though the anchoring layer. The domain layer also contains sets of tuples that describe the relation between concepts.

The user model in AHAM expresses individual user data as attributes of concepts using the object-oriented convention of `<object>.<property>=<value>`. For example, the following statement describes a user having learned the concept 'JavaPrograming':-

$$JavaProgramming.learned = true$$

Other UM information, such as background knowledge, preferences, age, etc., can also be stored in AHAM in a similar manner by treating this data as additional

attributes. To facilitate communication between AH systems, AHAM also defines two externally accessible accessor functions, getValue() and setValue(), for exchanging user model information. However it is acknowledged that such exchanges require a semantic understanding of the data; an issue to which at present AHAM offers no solution.

The adaptation model employed by AHAM, originally referred to as the teaching model, is represented as a set of event-driven adaptive rules, expressed as tuples, that change variables within the system depending on a user's actions. For example, a rule could be written that would set the user's 'read' field for a given concept to true when the user visits that concept.

$$\langle access(C) \Rightarrow C.read = true, post, true \rangle$$

The second component of this tuple indicates the phases for that rule: a *pre* phase indicates the rule is executed on the current user model before responding to the user, and a *post* phase occurs after a user has followed a link. The last component of the tuple is a propagate flag that indicates whether the triggering of one rule can propagate to other rules.

The final component of the AHAM model also found in the DEXTER model is the presentation specification, which contains a collection of rules in the same format as those used in the adaptation model. These presentation rules can set the value of presentation attributes to, among other values, GOOD, BAD or NEUTRAL depending on the appropriateness of the concept or link for the current user.

Using these modules, AHAM specifies an *adaptation engine* the role of which is to retrieve the user model, instantiate the adaptation rules using the user model and then execute all *pre* phase rules. The engine builds up the page using the presentation rules and displays it to the user in their browser window. On selection of a link, the post rules are triggered and the user model is updated with the new values.

Using these components, any adaptive hypermedia system (AHS) can be defined in AHAM as a 4-tuple of the above components.

$$AHS = \langle DomainModel, UserModel, AdaptationModel, AdaptiveEngine \rangle$$

AHAM is not the only model based on Dexter. The Munich Reference Model (Koch & Wirsing, 2002)(Koch & Wirsing, 2001) follows the same approach taken in AHAM by extending Dexter's storage model and adding user and adaptive models. Architecturally, the two reference models are almost identical. The major difference between the two approaches is that AHAM specifies an adaptation rule language, while the Munich model contains an object-oriented specification written in the Unified Modelling Language (UML). UML provides a visualisation of the reference model that can be supplemented with formal semantic information. The Munich model forms the basis of UML-based Web Engineering (UWE) (Koch & Kraus, 2002), an extension to UML that supports application design and lifecycle development. UWE guides the programmer through the process of developing an adaptive application, from requirements analysis, scenario development, task modelling and design to limited automatic code generation.

2.7.2 XAHM

The proposed XML Adaptive Hypermedia Model (XAHM) (Cannataro & Pugliese, 2001)(Cannataro, *et al.*, 2002) is a XML-based object-oriented model for AHS. XAHM defines a multidimensional approach, where each part of the AHS is described along three different "adaptivity dimensions" within an adaptation space.

- *User's Behaviour* - Browsing activities, preferences, etc.
- *External Environment* - Factors such as location (temporal, spatial), language, politics, etc.
- *Technology* - Network bandwidth, quality of service, etc.

These three dimensions form the space within which the user is tracked. The user model is generated using a mapping algorithm which maps the values in each of the three dimensions onto a stereotype user model. As the user's dimensionality changes, the system dynamically attempts to assign the user to the best-fitting stereotype group.

The *application domain model* is a layered model which represents the use of XAHM for a specific domain.

1. At the basic level there are *information fragments* consisting of text, sounds, images or videos. This data can be stored in any representation, structured or unstructured, and each is accompanied by an XML metadata element.
2. *Presentation descriptions* specify the layout and format of specific information fragments. These descriptions are specified by Document Type Definition (DTD)-constrained XML documents.
3. *Elementary abstract concepts* combine presentation descriptions to represent larger units of information. Represented as a directed graph (a digraph), the abstract concepts relate together presentation descriptions by weighted links. The arcs represent relationships between concepts, while the weights represent their relevance with respect to each other.
4. The top level *Adaptive Hypermedia* represents the elementary abstract concepts organised as a digraph. Arcs represent relationships between elementary abstract concepts, distinguished by the user's behaviour dimension. In the Adaptive Hypermedia, the arcs have no weighting attached to them.

The run-time system component of XAHM follows a three-tier architecture comprising of a *presentation layer*, an *application layer* and a *data layer*. The presentation layer receives and displays the pages. The application layer consists of a user modelling component, which calculates the user's position in the adaptation space and applies the mapping algorithm to determine the stereotype model, and an adaptive hypermedia application server, which extracts the presentation description and applies the logics contained within the stereotype model. Finally, the data layer stores persistent data such as information fragments and user data, and provides access methods to this data.

2.7.3 ELM-ART

The Episodic Learner Model-Adaptive Remote Tutor (ELM-ART) (Weber & Möllenberg, 1994)(Brusilovsky *et al.*, 1996a) is described as an adaptive, intelligent, web-based educational system. It is based on the earlier ELM-PE system that taught a Lisp programming course to university students and provided an Intelligent Learning

Environment with example-based programming. In 1996, the user base of ELM-PE was being limited by the inherent platform-dependent user interface and large processor requirements (Weber & Brusilovsky, 2001). These issues caused the developers to first translate the course material into Web pages and then port the functionality of ELM-PE to the Web, forming EML-ART in the process. Now the application has been generalised into a framework which has spawned several related adaptive applications.

Experience with ELM-ART and the related system INTERBOOK, highlighted areas for improvement particularly in regard to the restricted user model, basic adaptive features and issues involved with transferring a sequential book into an electronic hyperbook. This lead to the new enhanced ELM-ART II (Weber & Specht, 1997) that supported student-tutor and student-student communication, online exercises and tests, and a more expressive user model which provides greater feedback on previously visited concepts. The latest version of ELM-ART has been used as the basis for NetCoach (Weber *et al.,* 2001), an authoring tool to develop web-based courses.

ELM-ART stores a domain model containing units organised hierarchically into lessons, sections, subsections, and terminal pages. Terminal pages are shown at the end of units and can present the user with tests to work on, problem-solving examples or new concepts. Each unit contains information such as the main page text, associations with other concepts, concepts learnt by the user after completing the unit and test items. After the user has studied the unit and successfully answered the relevant test items, the user model is updated with the new learned concepts.

The user model also records the approach taken by students in solving programming problems posed by the system. These problems map onto the domain concepts of the course, and hierarchical *generalizations* are used to interpret student code. These generalizations contain plans and data pertaining to a particular aspect of the domain which ELM-ART uses to analyse the results of examinations. The complete student model (both the domain concepts learnt and approach to problems) can be used by ELM-ART to diagnose incorrect and incomplete student programs.

The latest version of ELM-ART has many adaptive features. It extends the link annotated traffic light metaphor originally developed by ELM-ART and employed in

INTERBOOK (shown in Figure 2-3) with an extended metaphor that uses white balls to indicate solved problems or visited pages, and orange balls placed next to uncompleted exercises or concepts that the system infers are known to the user. There is additional support for both direct guidance and exploratory learning and ELM-ART provides adaptive link ordering and adaptive testing using the information contained in the user model.

While being limited to courseware environments, ELM-ART has had much success in that area and there have been empirical studies analysing various aspects of the system, such as the direct guidance employed by ELM-ART II (Weber & Specht, 1997). ELM-ART's strength as a framework lies in the applications it has inspired, such as INTERBOOK (Section 2.6.1), ACE (Section 2.7.4), NetCoach (Weber *et al.*, 2001) and the various incarnations of the ELM system itself.

2.7.4 ACE

Motivated by work on the earlier system ELM-ART II, Adaptive Courseware Environment (ACE) has been developed as a Web-based environment to support adaptive learning (Specht & Oppermann, 1998). The approach taken in ACE is from a knowledge representation perspective, which has led to the modelling of three areas of knowledge implicit to the psychology of learning.

Domain Expert Model - A representation of the course material. This consists of a database of structured information, known as a knowledgebase, containing concepts (units), interrelations and dependences. Associated with each concept are learning materials such as examples, demonstrations, interactive playgrounds and tests. The author of this model can also cluster the concepts using an interest field, and specify curriculum sequence relations, which act as a default guide through the domain.

Pedagogical Expert Model - Contained in this module are the strategies and techniques related to teaching. By following the techniques of real teachers, specifically examining how they switch approaches when explaining a difficult concept or aiding a particular student, these techniques can be expressed as a set of strategies, such as Learn-by-doing, learn-by-reading or learn-by-example. A particular rule might check if a student has not seen a pre-requisite concept and if so warn the user and provide a link

to this concept. The Pedagogical Model in ACE also holds diagnostic information describing how to interpret test results and how such results impact on the various concepts related to the test (similar to the generalisations using in ELM-ART).

Learner Model - This model stores the learner settings, knowledge model and an interests model. The learner settings store profile information obtained from new users through direct questioning to obtain background knowledge, preferences and goals. The knowledge model contains a list of units the user has worked on; both completed and partially learnt. Finally, the interests model stores information relating to a user's interest in various concept clusters. If the user requests a given percentage of units from a cluster, the system will recommend further units from this cluster.

Using these three knowledge models, ACE can provide a variety of adaptive techniques. Adaptive link annotation, in the form of links marked as 'visited', 'not ready to be learnt' or 'recommended', is supported using information in the domain model. This model also provides direct guidance using the curriculum sequences defined by the author. This form of adaptive sequencing acts both at a course level, automatically selecting the next unit to visit, as well as at a unit level by sequencing the media within the unit. Finally, the system can also adaptively select a teaching strategy, overriding the default strategy, based on the concept material and key user features gained from the learner model.

Analysing ACE as a framework it is clear that the three knowledge modules provide a good level of abstraction away from the information domain. ACE has implemented a range of adaptive features drawing from both adaptive navigation and adaptive presentation technologies. ACE has been successfully implemented in two different courses; statistics (Specht *et al.*, 1997) and narcotics (Schoech *et al.*, 1998). The latest implementation includes an editor allowing an easy method for authors to create and modify courseware. Authors have an added ability to specify rules for adaptive instruction within their curricula, thereby giving authors some level of control over the adaptation provided by the system.

2.7.5 MetaLinks

MetaLinks (Murray, 2002) is a complete system for authoring and presenting adaptive hypermedia books (hyperbooks), closely related to previous systems such as INTERBOOK (Section 2.6.1) and ELM-ART (Section 2.7.3). Its application is aimed at large sources of interconnected online material. MetaLinks stores this material in a relational database, and authors use the *FileMaker Pro* software package to enter and structure the material. All content is centred on the chapter, section and subsection model. Authors can choose different styles and layouts for the presentation of the pages. Users connect to the web server through a standard browser and the system constructs the pages on the fly.

MetaLinks provides a range of adaptive features to the user. An annotated table of contents gives an overview of the location of the current page, in addition to a pictorial representation (map adaptation). To avoid cognitive overload, non-essential information such as examples, footnotes and glossary items, are hidden inside stretch-text that pops up whenever a user selects an associated word. Direct guidance is given through 'Next' and 'Explain more' buttons. The 'Next' button directs the users to the next sibling page, while the 'Explain more' button, descends a level and iterates over the child nodes.

MetaLinks also implements 'narrative smoothing' whereby each concept has an introductory piece of text associated with it. Whenever a user jumps to a page in a non-standard way, for instance if they used the search facility, this introductory text is pre-pended to the page. If the user arrives at a page by using the direct guidance buttons, then only the main body of the text will be displayed.

MetaLinks has been used to write four hyperbooks, the largest of which contains 400 pages and 320 glossary items. The domains included introductory geology, a MetaLinks user guide, 'Famous Woman in Mathematics' aimed at school children and a hyperbook about 'Early 20th Century Children's Games' written in collaboration with senior citizens.

As a framework, MetaLinks provides a single layer of abstraction above the information domain. The adaptive features, while broad ranging, are strongly integrated

into the system making them difficult to modify. The user model currently only stores implicit user information such as their trails, which includes how they arrived at each page, and the time spent on each page. MetaLinks' strongest feature is its support for authors, but there are still restrictions imposed on the authors, such as the choice about which types of adapted features are to be used.

2.8 Reflections on Adaptive Hypermedia Frameworks

2.8.1 Knowledge Modelling

All the systems presented here refer to both a Domain Model, which contains domain information (concepts) and relationships between aspects of the domain (concept relationships), and a User Model for capturing user-centred information. While the specific implementations of each model differ from framework to framework, conceptually they are identical.

ACE chooses to extend this view by including a Pedagogical Model, which captures knowledge about the various approaches to teaching the material. The Pedagogical Model in ACE contains teaching strategies for specifying actions in given circumstances and diagnostic information for interpreting test results. Pedagogical Models are a result of the AH community's rich history with intelligent tutoring systems; however, they do not translate well to non-education domains due to their instruction-orientated approach. An alternative would be to use a more generic model containing the rules for presenting information to users. Such a presentation model when used in an education domain would also include rules for student learning. An example of this more generalised type of model can be found in AHAM's Adaptation Model.

These data models provide a layer of abstraction above that of system implementation. Domain models maintain an element of independence from the information domain, allowing systems to operate across a variety of domains. User models impose a structure on user data facilitating the communication of user information between systems. Pedagogical Models provide designers with a simple way to explore a variety of different teaching and learning strategies.

In addition to domain, user and adaptation models, AHAM and XAHM also point to the need for an explicit adaptation engine whose job is to combine these three models together before applying the presentation specifications to build the page and then sending this data back to the user. In each of the other frameworks, the adaptation engine is not formalized but integrated into the entire system.

2.8.2 Modularisation: The future of application development

Looking at the history of programming, from early assembler languages through to higher level languages such as C & C++, component programming, object-oriented programming and now multi-agent systems (see Chapter 4) it is apparent that programming methodologies and designs are becoming increasingly modular. Whereas entire programs would exist in a single source file, new approaches to programming reduce programs into modules, then objects and now agents; each cooperating to achieve their own set of goals. The trends have led to more modular and autonomous approaches to programming, to ease the task of the programmer, increase platform independence and increase interoperability between systems. Such programming methodologies reflect the encapsulation, abstraction and self-contained nature associated with the various data models used in the AH frameworks.

2.8.3 The need to overcome communication difficulties

One of the issues raised by AHAM is the need for an agreement between systems if they are to request elements from each other's user model. Although there has been little research into this area of interoperability between adaptive hypermedia systems, other fields, most noticeably that of open hypermedia, have studied the effects of interoperability. If such a shared understanding is not present, then there is the potential for misinterpretation of information or misuse of personal user data by other applications. One solution to this problem is to allow each AHS to use their own terminology and then create an interchange format that would specify the mapping between the different terms used by each system. While an interchange format could work, a better solution in terms of interoperability would be to enforce a standard model for all AHS that specifies all the concepts, terms, objects and data types that an

AHS can use. Currently AHAM proposes the only standard for such large-scale interoperability, although only AHA! has adopted AHAM's approach.

2.9 Summary

This chapter has introduced the field of adaptive hypermedia, examining the history, terminology and technologies of the field. As the numbers of adaptive hypermedia applications continue to expand and diverge, researchers are starting to examine formal models and practical frameworks for developing AH. This chapter has also investigated these approaches.

The next chapter focuses on open hypermedia research, an area like adaptive hypermedia that has evolved from hypertext research in the late 1980's. While adaptive hypermedia research employed traditional engineering approaches to developing and producing adaptive hypertext, the open hypermedia community took a step back and examined ways of improving hypertext environments and attempted to define standards for increasing the interoperability between systems.

Chapter 3

Open Hypermedia

3.1 Introduction

This chapter presents the field of Open Hypermedia (OH), an area with many concepts, philosophies and models that have influenced this thesis. This is in part due to the Microcosm system (Section 3.2.3) which formed one of the early open hypermedia systems and in retrospect was based around an architecture similar in nature to that of agents. Although the system was primarily an open hypermedia system, it had many parallels with modern adaptive hypermedia applications.

This chapter details the evolution of open hypermedia research, originating from the hypertext community in the late 1980's, through the development of link services and onto the design of protocols for hypermedia systems. This chapter concludes with a description of the latest Southampton work into the Fundamental Open Hypermedia Model (FOHM) and contextual link server, Auld Linky.

3.2 History

Open hypermedia research, formed by a subgroup of the hypertext community in the early 1990's, originated with the idea that hypermedia functionality should be made accessible across all applications on a user's desktop (Davis *et al.*, 1992). This required the separation of links from documents and treating them as first class objects. These Open Hypermedia Systems (OHS) would store links in link databases (linkbases) thereby providing a centralised location for links from where they could be maintained, processed and propagated to other systems more simply than if they had been embedded into existing documents. This philosophy is also applied to the documents themselves, which can be accessed via external programs facilitating extensibility and

interoperability. It is these features that make open hypermedia systems ideal candidates for building distributed, scalable hypermedia systems (Kappel *et al.,* 1996).

3.2.1 Intermedia

One of the earliest systems to adopt this idea of a link service was developed at Brown University's Institute for Research in Information and Scholarship (IRIS). Intermedia (Yankelovich *et al.,* 1988) was a collection of desktop applications, text editors, graphics editors and 3D object viewers that collectively shared a common set of links maintained independently of each application. Users could author and follow links across these various applications. Intermedia supported bidirectional linking across ranges (or blocks) of text, which allowed authors to specify an area as an anchor, ranging from a single insertion point up to an entire document. Links were maintained in the Intermedia environment by a link server which also supported the concept of multiple linkbases (referred to as webs by Intermedia). These linkbases allowed users to impose a different set of links on the same document and also create and maintain their own separate set of links. Intermedia's only shortcoming as an open hypermedia environment is that it has a closed link service - one that can not be accessed or manipulated by the outside environment.

3.2.2 Sun's Link Service

Sun Microsystems' Link Service (Pearl, 1989) defined the first protocol for link management in OHS. Distributed as part of Sun's Network Software Environment (NSE), the link service consists of a single linkbase and a set of library functions for incorporating standardised hypertext functionality into applications developed on NSE. The link service acts as a central broker. Applications register their link handling capabilities with the service when they start executing. Like Intermedia before it, Sun's link service provided bidirectional links across applications, however Sun took a simplistic approach and assumed each anchor point was a single line of text. This design decision was taken as a means to simplify the problem of link maintenance after a file has been heavily edited. Deleted links are identified either on first activation of the link or by calling a garbage collection algorithm which verifies link integrity.

3.2.3 Microcosm

In 1989, work started at Southampton University to develop an open model for hypermedia, an approach which resulted in the Microcosm system (Fountain *et al.*, 1990). Microcosm tackled several problems that had been identified with existing hypermedia systems. Existing hypermedia systems required a large authoring effort, had a lack of communication with external software, used proprietary document formats, and imposed a distinction between authors and readers. To combat these issues, Microcosm was designed as an OHS which would provide users with dynamic, cross-application hyperlinks. Users could author their own hyperlinks across a multitude of document formats and applications and keep these links separately in linkbases managed by the Microcosm system.

In 1992, Microcosm was enhanced in several regards (Davis *et al., 1992*). The linking mechanism was extended to include a richer set of link types. *Specific* links originate from an object at a specific point in the source document, *local* links originate from an object at any point (such as a keyword) in a specific document, and *generic* links link from an object at any position in any document. Both the start and destination anchors of links could be anchored to a particular word, an offset within the document or to the entire document.

Three types of document viewers, or applications, could be used with Microcosm. Fully aware Microcosm viewers were applications written specifically for Microcosm, partially aware viewers consisted of third-party applications (such as Microsoft Word) for which a plug-in had been written to allow it to use the Microcosm functionality. Unaware viewers had no means of interfacing with Microcosm but they still could be used as link destinations.

Microcosm applications used an event-driven message-based interface. When the user clicked on a hyperlink or chose to follow links from a selected range of text in one of the Microcosm-aware viewers, this action would cause a message to be sent to the filter management system; a container that consisted of a series of filters (small processing units) arranged in a chain. The filter manager process would pass the message though the filter chain allowing each filter to process the message and attach new information onto the message. Filters could be programmed to suit a variety of

needs, such as searching for link destinations, add navigation tools or keep records of recent visited pages. They could even be used to store user models. Microcosm users were given the option of choosing filters dynamically at run-time by adding or removing them from the process chain (Hill *et al.*, 1993).

The modular design approach and extensive configurability of Microcosm allowed it to be easily and dynamically extended. Subsequent research looked into producing a distributed version based on a TCP/IP message passing protocol that would support multiple users (Hill & Hall, 1994).

3.2.4 DLS

One of the follow up projects to Microcosm, the Distributed Link Service (DLS) (Carr *et al.*, 1995), is a link delivery system that brings the concepts from the open hypermedia community to the wider Web audience. The DLS consisted of two parts: the link server, which consisted of a set of CGI scripts residing on a web server and accessed via HyperText Transfer Protocol (HTTP) requests, and the client interface; a plug-in written for Web browsers which provided the querying mechanisms to the link server. Unlike Microcosm, where viewer and link server ran on the same machine, the DLS supported multiple users operating in a fully distributed environment.

The server contained scripts for the creation, traversal and editing of links. The scripts had access to a variety of ‘context’ linkbases, configurable by the user. These hand authored linkbases provided specific links for a range of purposes. Linkbases might provide mappings from employee name to homepage in a company context, or a range of linkbases could be used to represent different user knowledge levels, contributed by an expert in a particular discipline. Users were also allocated their own personal linkbase for storing the links they create.

The client interface resembled the partially-aware Microcosm viewers. A plug-in provided existing web browsers with a dropdown menu bar containing options to follow links, create start and end points, edit links and change contexts. Links could be augmented with the existing page or users could select a range of text and issue the follow links request to the link server. Depending on the current active linkbases the link server would return with an intermediate page consisting of all the related links.

More recent work on the DLS has examined the use of software agent technology to realise the link server architecture (Carr *at al.*, 1998), although the definition of what constitutes and agent is very simplistic. In this approach, a single agent provides asynchronous link processing, thereby increasing the response of the system as links can be provided to the user after the original page has been displayed.

3.2.5 Chimera

Providing open hypertext principles to heterogeneous software environments is the objective of Chimera, a prototype hypertext system developed in 1994 at the University of California, Irvine (Anderson *et al.*, 1994). Software development environments develop and maintain a highly interrelated collection of software objects using a variety of tools and applications. In this type of environment, objects can have multiple views, for example, a cell in a spreadsheet can be displayed as a tabular collection of values or it can be represented as a pie chart, bar graph or any number of visualisations. The philosophy behind Chimera is aimed at providing these viewers with hypertext functionality, allowing the creation, manipulation and traversal of links seamlessly between both objects and viewers.

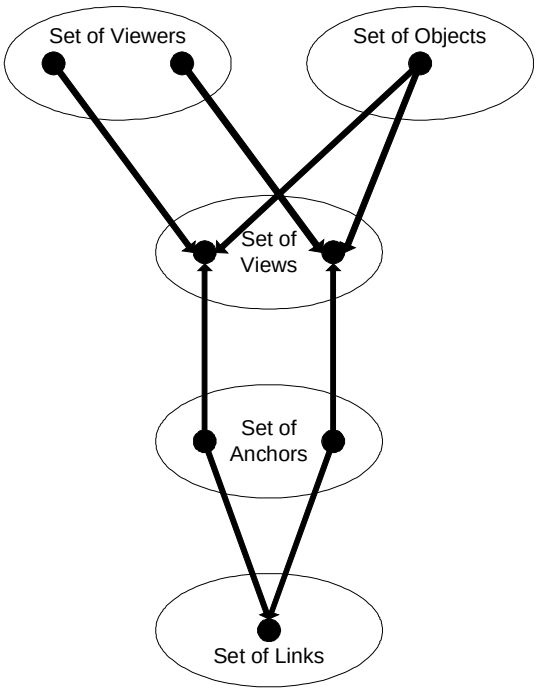

Figure 3-1. Chimera's hypertext concepts (Anderson *et al.*, 1994)

At the heart of Chimera is the notion of a view, which stores a mapping between an object and a viewer, see Figure 3-1. Objects are persistent entities and viewers are programs that display these objects. Because many viewers can display the same object, each object can therefore have multiple views. Anchors reference a portion of a view and Links collect together a set of anchors. Chimera stores these *n*-ary links as first class objects, which can themselves be viewed and even linked to.

The Chimera environment operates a client-server approach. Clients contain one or more viewers, and communicate hypertext events such as link traversal, anchor creation or link modification, with the server through an underlying API. The Chimera server principally provides hypertext services to clients by responding to and routing event messages from clients. The server also maintains a list of clients and active viewers that are ready to respond to hypertext events. Following the open hypermedia principle, anchors and objects are stored and maintained separately, with the anchor and link information managed by the server, while objects and views are handled by the Chimera clients.

Unlike Microcosm, which promotes a common interface style between Microcosm-aware applications, Chimera opts to leave the hypertext interaction style up to each viewer, which needs only conform to a low-level API. The two systems also differ in Chimera's use of the view as a means of abstracting objects from the anchor and linking mechanisms.

3.2.6 DHM

The DeVise Hypermedia (DHM) (Grønbæk & Trigg, 1994) system is an object-oriented approach to developing an open hypermedia system based upon the Dexter reference model (introduced in Section 2.7.1). Researchers at the University of Aarhus, Denmark, wanted to explore the features and limitations of Dexter within an open hypermedia environment. Dexter components are stored in an Object-Oriented DataBase (OODB), which can be shared between hypertexts. DHM follows a client-server architecture with runtime processes managing the within-component and runtime layers and connecting to the OODB server to transfer data. DHM provided a platform independent architecture through the development of different runtime processes that

would run on either Windows or Macintosh operating systems. Closed applications sit on top of these runtime processes providing the users with display and editing functionality for hypertext objects.

In the development of DHM, the researchers made several major modifications to the Dexter model. The first change was the inclusion of support for dangling links (links with zero or one endpoints). The DHM authors found that dangling links have many uses as they allow the creation of links in stages and proved useful in creating links to destination objects that had yet to exist. DHM also extended the notion of link directionality that exists in Dexter. Developers identified three types of directionality.

- *Semantic* - Specifies the relationship between endpoints.
- *Creation* - Determines the order the endpoint objects were created
- *Traversal* - Contains the methods of traversing each link (source $\rightarrow$ destination, bi-directional).

Heavy emphasis was also placed on the support for collaboration (Grønbæk *et al.*, 1993) and several different working practices were identified. DHM supported the sharing and collaboration in both synchronous and asynchronous modes by locking and notification mechanisms provided by the OODB and handled by all runtime processes. A locking mechanism was employed for environments where multiple clients attempt to simultaneously access data objects, while event notification provided support for asynchronous modes of cooperation.

3.3 Protocols for Open Hypermedia: OHP, OHP-NAV and FOHM

Over the decade of open hypermedia research, there has been an effort to define a generalised protocol for expressing hypertext objects with a view to providing a language to facilitate interoperability between systems. The first draft Open Hypermedia Protocol (OHP) (Davis *et al.*, 1996) proposal, developed in 1996 soon underwent radical changes. The suggestion that the protocol should cover other domains of hypertext systems, such as spatial and taxonomic, caused a rethink within the community and the core features present in OHP were renamed OHP-NAV to emphasize the navigational nature of the protocol (Millard, 2000).

Central to the OHP-NAV data model are the concepts of a *Node*, *DataRef*, *End Point* and *Link* shown in Figure 3-2.

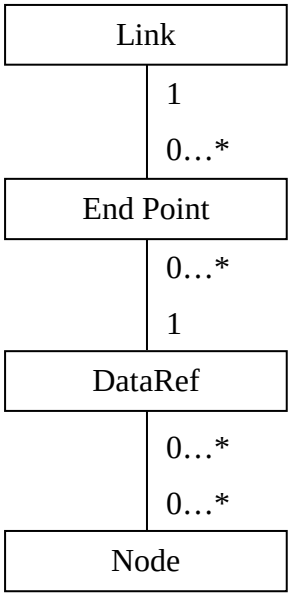

Figure 3-2. The Hypertext Data Model

- *Node* - A Node contains an item of information. Nodes can encapsulate words, paragraphs, concepts or entire documents. OHP-NAV does not specify the format of this data and consequently nodes can contain media of any type.
- *DataRef* - A DataRef is simply a pointer to the unique identifier of a particular data item, or a region within a data item. DataRefs provide a level of abstraction away from the data allowing structure and data to be maintained independently.
- *End Point* - An End Point is a container for specifying the attributes of an end of a link. End points most often contain direction attributes, such as source, destination or bi-directional, and these determine the possible actions that can be carried out on a link.
- *Link* - A Link conceptually represents some form of association between information. The link forms a relationship between DataRefs such as a simple source $\rightarrow$ destination navigational hyperlink, or *n*-ary links with multiple end points. Rather than referencing data items specifically, links point to end points which themselves locate the actual data items.

Following the development of OHP-NAV, researchers at the University of Southampton re-examined the protocol and formed a new model that would cover both taxonomic and spatial hypertext systems, as well as the more common navigational hypertext systems.

In taxonomic hypertext, information is categorised and placed in hierarchies forming taxonomies. Hypertext techniques such as non-linear information access and personalised views are used to locate and view this information.

Spatial hypertext is concerned with the organisation of information into logical spaces. Users navigate the domain by traversing the spaces themselves rather than the ad-hoc jumping in and out of areas that standard navigational hypertext systems allow. Spatial hypertext systems employ visualisations of the domain, presenting information nodes that are clustered, coloured and sized according to their relationships with existing nodes.

This work at Southampton resulted in the Fundamental Open Hypermedia Model (FOHM), see Figure 3-3. Unlike the protocols OHP and OHP-NAV, the FOHM model only defines the structure of hypertext objects and not the semantics of the communications mechanism. To distinguish between FOHM and earlier protocols, and to highlight the greater flexibility of FOHM, Nodes have been renamed *Data Items*, DataRefs are *References*, End Points are now called *Bindings* and Links are referred to as *Associations*. Two additional modifier objects, *Context* and *Behaviour*, can be attached to the various parts of the FOHM structure. Context is used to specify the conditions in which certain objects are visible, while Behaviour objects inform the client applications handling the FOHM structures of actions to perform given certain event conditions (such as *on display*, or *on link traversal*).

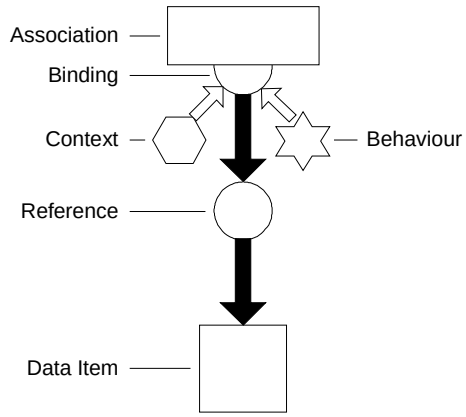

Figure 3-3. The essential components of a FOHM structure

FOHM draws on ideas developed in navigational, spatial and taxonomic hypertext systems in addition to the OHP-NAV work to provide interoperability between hypertext domains.

Firstly, FOHM provides the concept of linking to a reference rather than to the actual data item that can be seen in traditional navigational hypertext systems. References, previously called DataRefs in the hypertext model, facilitate linking to data regions, such as a 5-second segment of video footage or to a specific sentence within a document. References also point to associations, thereby providing a mechanism that allows complex structures of associations to be built.

Secondly, FOHM introduces the notion of a feature space composed of *featurevalue* objects to represent a reference's spatial aspects. This allows FOHM to classify links in a similar way to the method used by spatial hypertext systems. Featurevalues allow systems to gain an understanding about the type of association, expanding upon the traditional 'source', 'destination' or 'bi-directional' information and including metadata relating to the spatial features of the data items being referenced.

Finally, from the taxonomic hypertext domain, FOHM brings the notion of context to hypertext modelling. A *context* object in FOHM is a modifier that can be attached to associations, references or data items and used to specify which of those objects are visible given that the user is in a particular state. For example, a context object can be attached to a data item stating that the data is only viewable if the user's

context matches the context value ‘expert’. An alternative use of context might be attached to an association, which states that a particular video segment is only available to the user if they have sufficiently large bandwidth i.e. context = ‘ISDN | T1 | T3+’.

To explore the ideas behind FOHM, a contextual open hypermedia FOHM server was developed called Auld Linky (Michaelides *et al*., 2001). Auld Linky, which was originally called Auld Leaky, stores FOHM structures as XML objects and provides a mechanism for serving this structure to client applications. Clients query Auld Linky sending a FOHM structure and an associated context. The server uses a pattern matching process to compare the query structure against its internal FOHM structures and produce a set of matches. A culling process is applied to these results. Associations and data items with attached context that do not match the context supplied in the query are removed from the results. Before returning to the client, a final validity check is run on the structure to certify that the culling process did not destroy the validity of the original pattern match.

3.4 FOHM Structures

FOHM can be used to model many hypermedia structures. A recent paper has demonstrated that the implementation of almost all adaptive techniques present in Brusilovsky’s taxonomy (Figure 2-2) can be modelled using a small subset of the FOHM structures (Bailey *et al*., 2002). These structures are presented below.

3.4.1 Navigational Link

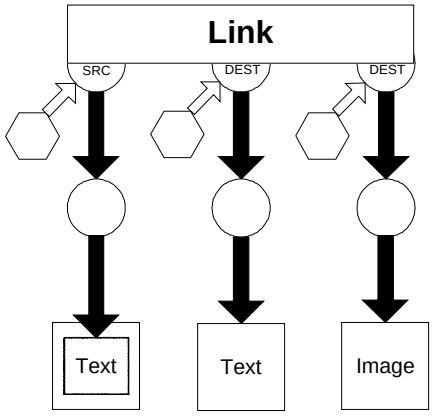

Figure 3-4. A FOHM Navigational Link

A navigational link is an association with typed data items; source, destination or bi-directional locations. Figure 3-4 contains a single source (SRC) anchor, and two destination (DEST) data items. FOHM also supports *regions*, allowing a start or destination point to refer to a specific location within a data item such as a word, paragraph, video segment or a region on an image. In Figure 3-4, the source location references a particular region in the object. The diagram also demonstrates context attached to the FHOM structure. Associations, bindings, references and data items can all have context attached.

3.4.2 Tour

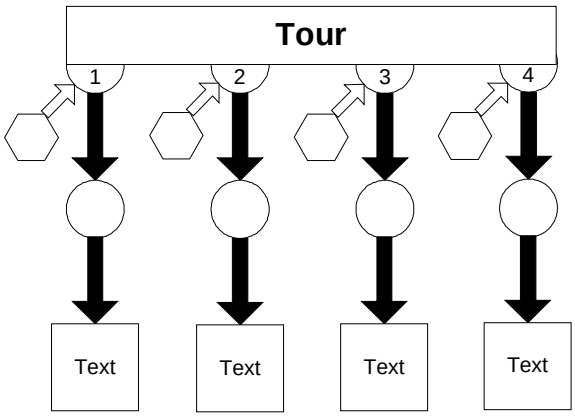

Figure 3-5. A FOHM Tour Structure

A tour is an association that represents an ordered set of objects, an example of which can be seen in Figure 3-5. The objects in a tour can be data items, other associations or a combination of the two. Tours can represent trails or paths through an information domain where a user would be presented with each item in sequence.

3.4.3 Level of Detail

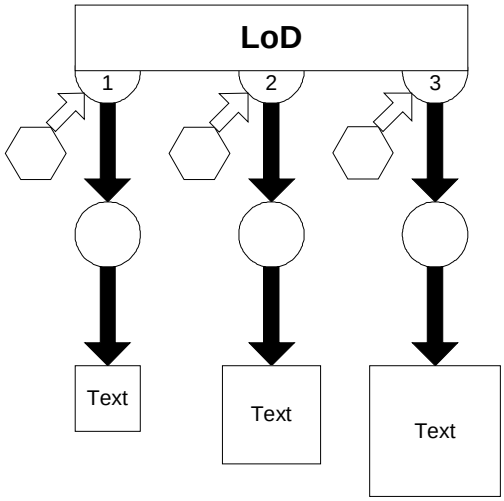

Figure 3-6. A FOHM Level of Detail Structure

A Level of Detail (LoD) structure, shown in Figure 3-6, represents an ordered sequence of objects, where each object presents the same conceptual information with increasing detail or complexity. For example, a LoD can be used to represent a concept where the first item is a brief summary, the second item is a short introduction and the third item a full description of the concept with additional examples and links to related information.

3.4.4 Concept

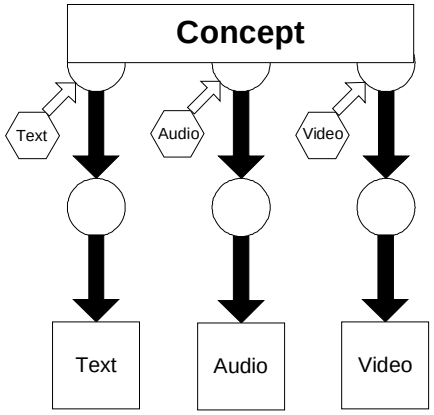

Figure 3-7. A FOHM Concept Structure

A concept is a collection of objects that represent the same conceptual information. An example of a concept can be seen in Figure 3-7. A concept can be used

to associate different media representations of the same information, such as text, audio or video, or collect together different methods of teaching the same information. In this instance, context objects are used to specify the media type. Users in an ‘audio’ context would only receive the audio fragment.

With the features and structures present in FOHM and implemented in Auld Linky it should be possible to model many adaptive hypermedia applications. For example, there is a similarity between the context culling mechanisms employed in Auld Linky for returning context-matching FOHM structures and the conditional if-then-else rules used in AHA! (Section 2.6.2). AHA! allows information fragments to be included or excluded depending on keywords in the user model. To support adaptive hypermedia, FOHM’s behaviour objects can be used as a mechanism for updating the user model by providing keywords, attached by on-display events. This specifies that the user has learnt a new concept when they view a new data item. This keyword knowledge, which is stored in the user model, can then be fed back into subsequent FOHM queries by attaching context modifiers specifying the new state of the user. This process is shown in Figure 3-8.

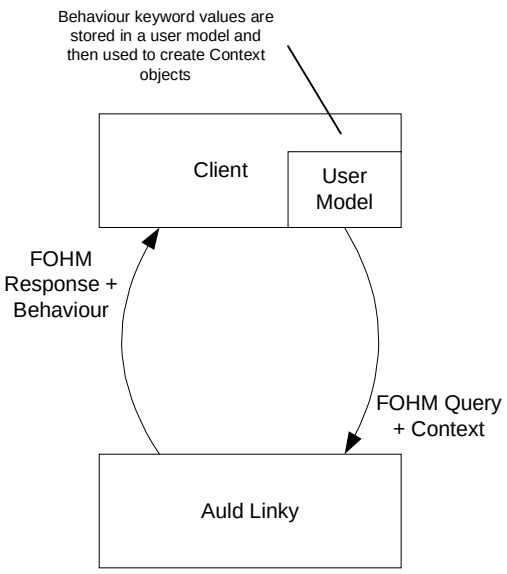

Figure 3-8. The process of User Modelling in Auld Linky

3.5 Summary

Open hypermedia research has changed quite significantly since its conception. This is partly due to the web’s increasing popularity as the hypertext environment of

choice by the masses, in spite of the many advanced features build into most OHS. Recent research has concentrated on developing protocols for increasing interoperability between open hypermedia systems, as well as models for expressing the domains of hypertext. This chapter has presented an overview of this work, focusing in depth on the more recent work with the Fundamental Open Hypermedia Model, and contextual link server; Auld Linky.

The next chapter examines the field of software agents, a popular and widely supported field which has seen much growth over the last decade.

Chapter 4

Software Agents

4.1 Introduction

The work described in this chapter draws heavily upon the use of software agents used mainly within agent-based systems. The field of agents has many diverse researchers, approaches and ideas, which help to create one of the more dynamic research areas in recent years. This chapter introduces the reader to the field of agents, looking at the history behind their development and the characteristics that help define modern software agents. The huge popularity of agent research has arisen at the time when object-oriented programming languages such as Java and C++ are proving such a success. This can be demonstrated by a quick visit to the popular search engine Google (Google, 2002) which will uncover over a hundred different agent frameworks (a listing of them can be found in Appendix B), of which this chapter will describe only a select few.

4.2 History

Agent research stems from the work in distributed artificial intelligence conducted in the 1970's. Carl Hewitt proposed an Actor system where each Actor had an explicit internal state and had the capability to respond to the messages of other Actors (Hewitt, 1977). While the subsequent years focused on the more theoretical aspect of bringing intelligence to software, and therefore agents, the last decade has seen a huge expansion of systems to solve practical problems (Nwana, 1996) drawing on advances in object-oriented programming, distributed processing, the Internet, the Web and the increased digitisation of information and services. The increased popularity of agents produced many different types of agents such as Belief Desire Intention (BDI) agents (Rao & Georgeff, 1991), weak/strong agents (Wooldridge &

Jennings, 1995), interface agents (Maes, 1994), distributed or multi-agent systems (Minar, 1998)(Flores-Mendez, 1999), autonomous agents (Franklin & Graesser, 1997) and even emotional agents (Bates, 1994). Complimenting this list is an equally extensive and varied set of domains that agents have been applied to; including military intelligence gathering, industrial production planning, network resource management, robotic football, aircraft maintenance tools, financial portfolio management; and a host of e-business and internet applications such as online auctions, web mining and comparison shopping.

4.3 Agent Characteristics

The influx of various approaches under the banner of 'agents', caused a need to classify and define this term. However it quickly became apparent that everyone had their own definition (Franklin & Graesser, 1997) due in part to the historical relationship with the AI community and the vague notion of intelligence (El-Beltagy, 2001). Of the many definitions for the nature of agents (often referred to as the term agenthood (Jennings, 2000)(Nwana, 1996)) that have been proposed, most have at their centre a set of defining characteristics that every agent must demonstrate. For instance BDI agents must show explicit beliefs (knowledge perceived to be true), desires (goals) and intentions (plan to obtain goals) (Rao & Georgeff, 1991), while Woodridge and Jennings' weak agents (Wooldridge & Jennings, 1995) should be autonomous, reactive and social. A third definition from (Franklin & Graesser, 1997) lists autonomous, reactive, communicative, adaptive, mobile, flexible, goal-oriented, continuous and with some form of character or emotion.

Using such definitions from the literature, it has been possible to create a set of primary, secondary and tertiary agent characteristics. Primary characteristics are inherent to most of the popular agent definitions, secondary characteristics are the set of extended characteristics usually associated with agents. The third set of characteristics contains more abstract, desirable, human-like features.

Primary Characteristics

- *Autonomous* - An agent should be able to execute without the need for human interaction, although intermittent interaction may be required.

- *Social / Communicative* - An agent should have a high level of communication with other agents. The most common protocol for agent communication is the Knowledge Query and Manipulation Language (KQML) (Finin *et al.*, 1997).
- *Reactive / Responsive* - An agent should be able to perceive its environment and react to changes in it.

Secondary Characteristics

- *Proactive* - Proactive agents do no just react to their environment but can take active steps to change that environment according to their own desires.
- *Adaptive* - Adaptive agents have the ability to adjust their behaviour over time in response to internal knowledge or changes in the environment around them.
- *Goal-oriented / Intentions* - These agents have an explicit internal plan of action to accomplish a goal or set of objectives.
- *Persistence / Continuous* - Persistent agents have an internal state that remains consistent over time.
- *Mobility* - Mobile agents can proactively decide to migrate to a different machine or network while maintaining persistence.

Tertiary Characteristics

- *Emotion* - Agents with the ability to express human-like emotion or mood. Such agents might also have some form of anthropomorphic character or appearance.
- *Intelligence* - Agents with the ability to reason, learn and adapt over time.
- *Honesty* - Agents that believe in the truthful nature of the information they pass on.

These agent characteristics lead to many advantageous features. The very nature of agents as independent, social entities that can respond to and change their environment provides a strong foundation for building reliable, robust, flexible, extensible and scalable systems. Agents can help ease user tasks (Maes, 1994)(Nwana, 1996) and adapt to user requirements (Moukas, 1996)(Nwana, 1996). In spite of their many benefits, agents are not the solution to every problem. One major disadvantage of building agent systems is that the complexity of agent interactions and dynamic nature of the agents themselves make it difficult to predict agent behaviour (Wagner, 2000). This can cause problems in safety critical environments where outcomes need to be

assured. However as the computer world is becoming increasingly networked and distributed, agents are likely to become the next engineering paradigm for system development. (Jennings, 2000).

4.4 Ontologies for Agents

Ontologies, a term borrowed from ancient Philosophy, originated in the knowledge acquisition field and was originally defined as an *"explicit specification of a conceptualisation"* (Gruber, 1993). Ontologies represent a world view in a particular domain which consists of concepts, definitions and concept-relationships. The term conceptualisation is described as "an intentional semantic structure, which encodes the implicit rules constraining the structure of a piece of reality" (Guarino & Giaretta, 1995).

Ontologies are useful because they facilitate a shared understanding in a particular domain of interest (Uschold & Gruninger, 1996). They have a wide range of applications including knowledge engineering, knowledge representation, database design, information modelling and agent-based system design (Guarino, 1998). Ontologies have been borrowed by the agent community because they provide a shared world view in which each agent can ground its beliefs and actions (Huhns & Singh, 1997).

Ontologies provide agents with a powerful domain of discourse. Before conversing, agents can agree to use a particular ontology, which defines some aspect of the world. Using such a shared ontology, agents can then discuss a topic with complete confidence that other agents have the same understanding of the items discussed. For instance, a mutually exchanged 'aeroplane' ontology allows one agent to talk about a Cessna 180, and for the other agents to understand that a Cessna 180 is a type of aeroplane object, which has several properties. In this instance, the Cessna 180 has a single propeller and seats 4-6 people.

4.5 Standardisation Efforts

There are currently a wide range of different agent architectures, frameworks and systems developed for both research and industrial purposes. To unify these approaches

three standardisation efforts have appeared with the overall aim of increasing interoperability between agent systems.

- MASIF - The Mobile Agent System Interoperability Facility (MASIF) (MASIF, 2002)(Milojicic *et al.,* 1998) has been in development by the Object Management Group (OMG) since 1995 to promote interoperability and mobility among agent platforms.
- KQML - The Knowledge Query Meta Language (KQML) (KQML, 2002)(Finin *et al.,* 1997) is one of the most popular and widely used protocols for defining agent-to-agent communication. KQML is the oldest project, developed in 1992 by the DARPA Knowledge Sharing Effort consortium.
- FIPA - The most recent addition is Foundation for Intelligent Physical Agents (FIPA) (FIPA, 2002)(O'Brien & Nicol, 1999), a non-profit organisation created in 1996 aimed at developing software standards for maximising interoperability within and across agent-based systems.

Of these three approaches, MASIF uses a procedure-oriented interaction model using Remote Procedure Calls (RPC) or Remote Method Invocation (RMI) technology, while both KQML and FIPA both specify a message-oriented Agent Communication Language (ACL). The ACL model used in both FIPA and KQML is based on speech act theory (Searle, 1969), a field of research aimed at analysing the semantic content of vocalised messages.

These standards facilitate agent interaction across hardware platforms, operating systems, programming languages and agent platforms. Recent FIPA compatibility tests (FIPA, 2001) have already shown successful interoperability through the transfer of ACL messages between several FIPA compliant frameworks.

4.6 Agent Frameworks

Due to the sheer numbers of agent frameworks available, an extensive analysis of them all would be out of the scope of this thesis. Instead, three systems will be examined in this section. Section 4.6.1 details JATLite, a collection of packages for simple KQML agent development. This is followed in Section 4.6.2 by JADE, a FIPA-

compliant agent framework and development kit. Finally Section 4.6.3 describes SoFAR, a research framework with a strong emphasis on knowledge modelling. On the most fundamental level, each framework supports three features for agent developers.

- Creation: Each framework provides the ability to quickly create and run agents within a supported environment.
- Communication: Each framework supports agent-to-agent communication using speech acts.
- Discovery: Each framework allows agents to find new agents using a service based discovery mechanism.

On top of this, each framework offers a unique set of additional features such as standards compliance, mobility, interoperability, knowledge-based ontologies, graphical interfaces etc.

4.6.1 JATLite

Originally created as an academic research project at Stanford University, the Java Agent Template, Lite (JATLite) (Jeon *et al.,* 2000) is a collection of Java packages designed to provide a communication and interaction mechanism to programs distributed across the internet. JATLite defines an agent as part of a community that perform a distributed computation using typed-messages. JATLite is aimed towards communities of 'dumb' agents rather than individual intelligent agents. It is possible to use JATLite agents to write wrapper classes around existing legacy software and allow them to interact across a network to other agents.

Aside from agent classes, JATLite provides a message routing and buffering service to its agents. Each agent keeps a record of the Agent Message Router, which acts like an email server and forwards messages onto the receiver. If a connection cannot be made with the receiver, the message router will buffer the message until the receiving agent resumes its contact with the router. The advantage of this mechanism is that agents are allowed to disconnect, migrate to a new location and then reconnect to the router to receive their messages. To prevent mobile agents from being impersonated, agents carry a password with them to identify themselves with the router. As the router keeps a directory of all registered agents, it provides a lookup

service to other agents who need only keep track of the system router rather than the locations of other agents, although using a router in this way does introduce a single point of failure into the system.

The ACL used by JATLite is a non-standard variant of KQML, which contains connection and identification methods, absent in the KQML specification. As JATLite has no notion of a distributed clock, the KQML messages are not guaranteed to arrive in the same order as they were sent. For extremely large information transfers, the system provides both file transfer (FTP) and email sending (SMTP) capabilities for its agents.

4.6.2 JADE

Java Agent DEvelopment Framework (JADE) (Bellifemine *et al*., 1999) is an open source Java implemented middleware framework that became one of the early systems to adopt the FIPA specifications and become FIPA compliant. It provides an efficient, scaleable, distributed environment that fully implements all aspects of the FIPA communications model.

JADE was designed to ease the process of developing agent applications and it uses several mechanisms for accomplishing this. JADE abstracts the communications infrastructure away from the programmer, providing transparent mechanisms for locating both local and remote agents and initiating conversations with them. The communications model used by JADE is FIPA's ACL Each JADE agent runs as a single thread within an agent container, and collections of containers run within a single JVM platform, see Figure 4-1. Each container has a dedicated directory facilitator that acts as a yellow pages, providing the other agents in the container with a means to look up agents based on the services they offer. The JADE platform also contains a front-end container with a graphical user interface agent that allows users to interact with the platform and provide a mechanism for controlling each agent. The interface provides methods for starting, suspending and stopping agents as well as debugging features such as activating a sniffer to intercept communication packets, and sending custom messages to agents.

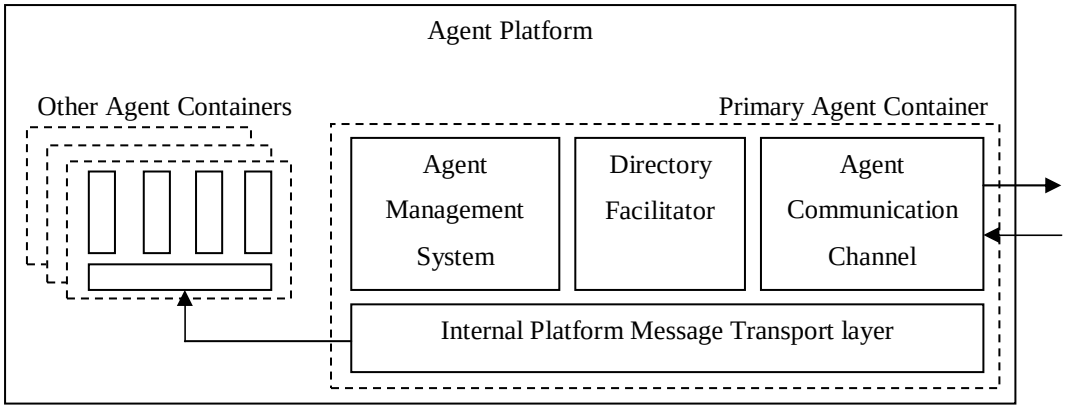

Figure 4-1. The FIPA reference model implemented by JADE

The distributed nature of JADE allows multiple JADA and other FIPA-compliant agent platforms to communicate via the Internet Inter-ORB Protocol (IIOP). IIOP is a message transport protocol, allowing objects to be sent across networks in a platform independent manner. An agent communication channel, which is physically distributed across all JADE platforms, provides agents with a transparent mechanism for converting incoming ACL messages from an external platform into the ACLMessage format used internally in each JADE platform and visa versa. Between containers, Java's Remote Method Invocation technology operates as the transport mechanism, and agents in the same container will communicate directly with each other. When an agent communicates with another agent, the JADE framework will select the most efficient communications method and therefore appears transparent to the agent and the agent's programmer. Tests of the JADE communications model have shown linear scalability, which mimics that of RMI when agents are distributed across a network (Vitaglione *et al.*, 2002).

The latest version of JADE conforms to the FIPA2000 specification and currently has many international companies, projects and universities using the framework. There is also a JessAgent written for JADE that allows programmers to take advantage of the full capabilities of the JESS expert system (JESS, 2002) within their programs.

4.6.3 SoFAR

The Southampton Framework for Agent Research (SoFAR) (Moreau *et al.,* 2000) was designed to provide a reactive environment for large numbers of Distributed

Information Management (DIM) agents communicating over an implementation-independent communications infrastructure.

DIM agents manage the entire life-cycle of information; from discovery and creation to management and integrity maintenance (Dale & De Roure, 1997). In an agent environment, there are large groups of DIM agents each managing a single aspect of the information life-cycle. Although each agent is conceptually quite simple with little knowledge of the world outside its assigned task, the agents working together produce the appearance of intelligence.

SoFAR provides a lightweight and scalable architecture for the deployment of agents. These agents run within a single JVM environment called a platform. Every platform contains a registry agent whose job is to maintain a list of services that each agent within the platform provides. Agents contact the registry, advertise the set of services they offer and give their location within the platform. If an agent requires the services of another agent, it must first query the registry with a request for the required service. The registry agent responds with the location of each agent that can handle the requested service. Now the first agent is free to contact each agent directly and initiate a conversation. To accomplish this, SoFAR agents rely heavily on the use of ontologies to exchange information. Ontologies provide a mechanism for defining a world view, allowing agents to agree on definitions of entities before the communication is initiated. Section 4.4 discusses the use of ontologies within agent environments.

SoFAR also supports mobile agents. If an agent cannot find a requested service but knows the location of a new platform, it can request that the current platform serialise it and transfer the agent to a new platform. When it arrives, the new platform will de-serialise and reactivate the agent. When the agent has registered itself with the new registry agent it is free to pursue its original goal. A detailed description of the migration service implemented by SoFAR is documented in (Moreau *et al.*, 2001).

The communication model within SoFAR is based around a core subset of functionality from existing agent communication languages such as FIPA and KQML. Agents within SoFAR can offer registration and subscription services, as well as information performatives such as request, assert and query. SoFAR hides the message

delivery mechanism within an abstract agent class from which all other agents are derived. This provides a layer of abstraction between the low-level communications infrastructure and the high-level XML-based agent description format SoFAR uses to define each agent.

With these services, SoFAR provides all the necessary infrastructure needed to rapidly develop autonomous, proactive and socially aware agents.

4.7 Summary

This chapter has presented a background to agent-based technology, examining the definitions, features and properties of agents, and looking at the recent efforts in defining standards for agent-based system interoperability. In light of the numerous number of agents frameworks available, three different frameworks have been examined in detail. Although each approach provides various advanced features, the review has shown that at their most basic level each framework enable developers to produce agent systems with the same fundamental properties.

The next chapter examines the field of recommender systems, one example of a research field which has successfully embraced agent technology.

Chapter 5

Recommender Systems

This chapter presents the field of recommender systems. Like adaptive hypermedia systems, recommender systems aim to minimise user disorientation in large digital resources by suggesting useful or interesting information to a user, based on either the recommendations of other users (collaborative recommendation), or the analysis of items already seen by the user (content-based recommendation). Unlike AH systems where it is possible to tailor both the content and navigation of a user's experience through a domain, recommender systems concentrate on selecting or filtering links, or information, which has specific interest to the user.

Following on from the agent discussion in the previous chapter, recommender systems provide one example of the widespread use of agent technology. Many collaborative recommender systems are built using agents, although relatively few use an existing standard agent framework. This means that not every 'agent-based' recommender system will adhere to the defining characteristics presented earlier in this chapter. However, recommender systems still provide a good example of how agent technology can easily be applied to adaptive situations and this contrasts with the adaptive hypermedia community, which has largely ignored the technology. This chapter will examine the various types of recommender systems in existence before presenting five example systems, each built using agent technology.

Recommendation services can be supplied in any situation where a large resource is being viewed by a great number of users; examples include electronic encyclopaedias, news resources, retail web sites or publication databases. The Web's popularity and size lends itself well as an environment for recommender systems, and as a result companies have been quick to embrace this technology (Schafer et al.,

1999). Examples of such companies include online book retailers Amazon (Amazon.com, 2002) and Barnes & Noble (Barnes & Noble, 2002) who have both incorporated a recommendation service for their users. More recently, search engine providers Lycos (Lycos, 2002) and Google (Google, 2002) have implemented tools allowing users to look at other people's comments and ratings about a web site.

5.1 Collaborative Recommender Systems

Collaborative recommender systems are socially aware systems that use the previous interactions of individual users to find similarities with the current user. At the heart of any collaborative recommender system are a set of user models, which contain a user's trail and a record of actions that the user has taken e.g. purchasing goods, requesting more information, sending emails etc. With these models it is possible to compare behaviour patterns and then group them using standard machine classification techniques. Individuals who match a specific group of users will be presented with links based on the places these group members have visited. This idea is based on the psychological theory that while two users might never have identical user models, social groups will form around users with similar interests (Pennock *et al.*, 2000).

However, one disadvantage of collaborative recommender systems is that they often require explicit user feedback. This produces problems as studies have shown that users are reluctant to provide any sort of conscious feedback without some form of incentive (Morita & Shinoda, 1994). This is particularly prevalent early on in system deployment as no recommendations can be given until users have first entered some ratings. This has become known as the cold start problem (Maltz & Ehrlich, 1995). Another disadvantage of these systems is that they can only suggest previously visited pages, and therefore designers have to engineer methods of pro-actively finding new resources and recommending them to their users. The third major problem with document recommendation occurs with certain individual users who have unique interests (Balabanović & Shoham, 1997). Their trails fail to match any other group of users and this leads to poor recommendations (Mooney & Roy, 2000). This problem can be overcome by increasing the number of users or by using an alternative system for recommendation.

5.2 Content-Based Recommendation Systems

With a content-based recommender system, the basis for recommendation comes from the content on the pages that a user has viewed and not the behaviour of other users. Each page is analysed using information retrieval (IR) techniques. IR attempts to identify features in a document to aid classification. These results are fed into the user model and stored as user interests. This removes the need for multiple users and allows content-based systems to start providing recommendations as soon as the user interacts with the system. An additional advantage of content-based recommender systems is that by identifying user interests this provides a very useful resource for other forms of adaptation to augment the recommendation service.

Content-based recommendation still produces many problems. Firstly, as time progresses, the system becomes too specialised in the user's interests and this makes it hard for the system to broaden its set of recommendations. Secondly, a content-based approach is unable to distinguish between high-quality and low-quality information and also ignores the aesthetic qualities of pages such as images, and download time, which can all influence a user's opinion of a page (Balabanović & Shoham, 1997)(Claypool *et al.*, 1999).

Because of the weaknesses of both content and collaborative recommendation techniques, some of the latest recommendation systems that have appeared in the latter half of the 1990's are drawing on both techniques to provide recommendations. These new hybrid recommender systems, such as the agent-based FAB (described in Section 5.3.5) and the fuzzy categorization of TalkMine (Rocha, 2001), can use the strengths of both techniques to overcome their individual weaknesses (Basu *et al.*, 1998)(Claypool *et al.*, 1999).

5.3 Examples of Recommender Systems

This section details a selection of the more innovative content-based, collaborative and hybrid recommender systems developed in the past decade. All recommender systems except WebMate use multiple agents as their underlying component architecture; demonstrating the use of agents for dynamic information environments. MEMOIR, Section 5.3.1, and the subsequent QuIC project, Section

5.3.2, provide examples of collaborative recommender systems. This is followed by two personal agent approaches which employ information retrieval techniques to increase search engine relevance in Personal WebWatcher, Section 5.3.3, or produce personalised newspapers in WebMate, Section 5.3.4. The final system, FAB, Section 5.3.2, uses a hybrid approach with content-based agents to actively search for new documents and collaborative personal agents that help identify groups of users with similar interests.

5.3.1 MEMOIR

MEMOIR (Pikrakis *et al.*, 1998) is an agent-based system, designed to support researchers working with vast quantities of distributed information in finding both relevant documents and other researchers with related interests. Although not developed as such, it can be viewed as a collaborative recommender system. MEMOIR finds related documents and people through a comparison of user trails, which the system regards as first class objects. There are two types of trails; user trails formed from the documents a user visits, and shared trails created by users who grouped related interesting documents together. If the current document appears in one of these shared trails, then the system makes recommendations to other documents in the trail. A proactive 'Similar User' agent informs the user of other users with similar interests by analysing the overlap of the current user trail with those of other users in the system (De Roure *et al.*, 1998).

MEMOIR is based around a two-tier agent architecture, with simple agents in the first layer to access databases and present information to the user, and more intelligent information broker agents in the second layer. These second layer agents allow the introduction of additional resource discovery agents to provide more specialised recommendations.

The experiences and techniques that arose from the MEMOIR project helped spur on the development of the agent-based recommender system QuIC.

5.3.2 QuIC

The Queries In Context (QuIC) system (El-Beltagy *et al.*, 2001)(El-Beltagy *et al.*, 1999) provides a collaborative recommender service using an agent-based distributed information management environment. The agent infrastructure is mainly Java-based, and uses KQML as the communications language. Agents work together within the system to support collaborative user queries and recommend links to users based on the current context of the document.

The central agent in QuIC is a directory service agent called the facilitator for registering services and routing messages. The facilitator supports the dynamic addition and retraction of services allowing for an extensible and open architecture. In addition to this, there is an organisational memory agent that records the URL's and bookmarks of the users. This agent is capable of responding to queries such as "*Who has seen the following URL*" and "*Recommend URLs related to this document*". The organisational memory agent is written in Prolog. QuIC also includes a Microcosm-inspired link service agent. This agent autonomously collects and maintains clusters of links, arranged by context. QuIC defines context as a feature vector of related terms; a collection of keywords that form a collective representation of the destination of each link. The link service agent receives a request for links containing a keyword or group of keywords and uses its own internal linkbase in addition to the services of other agents to compose a set of links that match the initial query.

Each user is assigned a personal user interface agent to interact with. This agent records information entered by the user such as their interests, contact info, web pages etc. Browsing history and bookmarked page information are presented to other agents upon request. The interface agent also provides the user with a query facility for interacting with other agents in the system. Responses are collated by the agent and presented to the user.

The user interface agent's main role is to act as a proxy, receiving page requests from the user's browser, fetching the page and storing a record of the page in a history list. The interface agent creates a representation of a documents as a [keyword,weight] document vector (also known as a feature vector). This lends to the definition of context in the QuIC system as a user's trail through these document vectors. To

produce a document vector, a three-stage algorithm using existing information retrieval techniques has been developed (El-Beltagy *et al.,* 2001).

1. Firstly, a web page is analysed and the stop-words are removed. Stop-words are the most frequently occurring words in a given language e.g.: ‘a’, ‘the’, ‘and’.
2. The remained words are then stemmed using the Porter stemming algorithm (Porter, 1980); a technique which reduces words to their root representation. For example keywords ‘sounding’, ‘sounded’ and ‘sounds’ would all become ‘sound’.
3. The final step applies a Term-Frequency, Inverse Document Frequency (TFIDF) algorithm to the text to produce a weighted set of keywords. TFIDF gives each word a value, or weight, depending on how often it occurs in the current document given the word’s likely occurrence in a random sample of documents. So for example, if the words ‘music’ and ‘Opera’ occurred the same number of times in the web page, the keyword ‘Opera’ would probably be given a higher weight as it appears less frequently in a random sample of English documents.

When the document vector has been created, the interface agent contacts the link service, sends the vector and requests a set of matching links for the given document context. The link service clusters each document’s vector using a similarity cosine function in combination with a heuristic function. The similarity cosine function measures the cosine angle between two vectors and this value (between 0 and 1) specifies how closely they match each another. After the cluster with the closest similarity with the document has been identified, links are found which match that cluster. These links are then inserted into the current document. Because the links originate from the cluster that matches the current feature vector, the links are all related to the current context of the document. In this way, QuIC contains the notion of context to further restrict the set of recommended links and provides a mechanism for linking in context.

5.3.3 Personal WebWatcher

Personal WebWatcher (PWW) (Mladenic, 1996), is inspired by earlier work on WebWatcher (Joachims *et al.,* 1997), which located information on the web and

presented it to users when they provided a search engine-like request in the form of a set of keywords. PWW is a personal agent content-based recommender system that accompanies a user from page to page as they browse the web. PWW highlights hyperlinks in a web page that it believes will be of interest to the user. The designers wanted to remove the need for users to enter explicit information about themselves as was required by WebWatcher, so instead during periods of reduced user activity, e.g. at night, PWW analyses the user's navigational trail and constructs the user model from this information. The content from these trails is processed to obtain a set of keywords. The aforementioned TFIDF algorithm is then applied to form a set of associated weights. Finally, these vectors of word-weight pairs are analysed using a Naïve Bayesian classifier algorithm to form a model of the user interests. The naïve Bayesian classifier (Langley *et al.*, 1992) is a modified version of Bayes' theorem where a simple probabilistic formula is used to form a probability given a set of incomplete data items and which can then update its probability when new information arrives.

PWW processes all the pages on a user's trail, and weights them as positive examples of the user's interests, while all the documents linked from these pages that do not appear on the trail, and which therefore the user did not visit, are assumed to be negative. The designer's reasoning is that all the links were presented to the user but they chose not to follow those links, which indicates a negative interest in the page. While the user browses the web, this model is used to give suggestions on which hyperlinks on the page the user might find interesting, by inserting small graphics next to the relevant links. Because downloading each hyper-linked page is a very slow process, especially for big documents with many links on, PWW's real-time link annotated recommendations are based on the text label of each link as it appears in the page.

While it seems that PWW is a relatively easy system to evaluate, when compared to some of the other recommender systems, the authors have since focused on analysing individual aspects of the PWW system. Specifically, several papers have been written about the feature selection and new document categorization mechanisms, but a result, no one has yet provided a complete overall evaluation of the accuracy of the Personal Web Watcher recommendation system.

5.3.4 WebMate

WebMate (Chen & Sycara, 1998), another content-based recommender system is a stand-alone proxy that monitors a user's web activity and uses an applet controller to act as a user interface to the proxy. Explicit feedback occurs whenever a user is interested in the page. They select an 'I like it' option in the controller applet and then WebMate utilizes the TFIDF algorithm to produce a weight vector for that document. Documents are categorized by applying a similarity function and a nearest neighbour algorithm to group similar documents together.

WebMate uses these categories to produce a personal newspaper for the user. To do this the user supplies a list of URL's, which point to on-line news resources such as those maintained by the BBC (BBC, 2002) or CNN (CNN, 2002). WebMate traverses the list and for each page it extracts the links in each headline and then constructs vectors for each of these new pages. For each of these vectors, a cosine similarity function is applied to the current profile and if it exceeds some threshold, then the page is added to the personal newspaper. As a final measure before it is presented to the user, adaptive link sorting is applied to each article in the newspaper based on the similarity with the user's profile.

When the relevance of each news article is measured, it was found to produce an accuracy rating of between 50%-60% for the top 10 ten news stories it suggests. When it was expanded to include more than 500 pieces of news a day, the accuracy of the system fell to around 30%. However, this is still better then the approximate accuracy of 20% for a randomly chosen set of news stories for which the authors guesstimate 1 out of every 5 articles may be of some interest.

WebMate also provides a keyword expansion algorithm based on the information contained in the user model to form a Trigger Pairs model (Gauch & Futrelle, 1994). To accomplish this, another information retrieval technique called Mutual Information (MI) (Church & Hanks, 1989), is used to analyse every word and its surrounding text to form a set of co-occurring words. This set contains the triggers for the corresponding triggered word. The MI algorithm is applied to all the pages that a user has marked as interesting. When a user enters a list of keywords into a search engine, WebMate grabs the keywords and searches the word database for matching triggered words. If any are

found then it appends the top n triggers for each match to the existing search query before submitting it. This is a relatively simple but powerful and unobtrusive mechanism for increasing the relevance of search engine results.

5.3.5 FAB

In development since 1994 as part of the Stanford University digital library project, FAB (Balabanović, 1997)(Balabanović & Shoham, 1997) is a recommender system which combines the advantageous elements of both collaborate and content-based recommendations techniques.

The FAB system is based around a two-stage document collection and then user selection process, and this is reflected in the underlying agent architecture. The agents in FAB are simple processes that keep a persistent record of their state and demonstrate many of the primary and secondary agent characteristics presented in Section 4.3. In the collection stage, documents are farmed from the web via a set of collection agents, each of which maintains a profile for a particular topic. Using the information retrieval TFIDF algorithm, keywords are harvested from the web pages forming a representation vector for that page. Each agent then employs a cosine function and periodically sends the pages that best match its topic to a central repository.

In the selection stage users receive web page recommendations on demand through a personal agent, which maintains a profile of the user's current interests. This user model is updated via relevance feedback given by the user whenever they have finished looking at a recommended web page. A seven-point scale is used to determine the user's rating of a web page and this is recorded in the personal agent's profile and forwarded on to the collection agents' topic profiles. In addition, any highly rated pages are also sent to other personal agents with similar user interests. In this way, groups of users can be identified by the system and modelled in the topics of the various collection agents, which adapt to entire groups rather than single individuals.

The use of both content-based and collaborative recommendations has many advantages. FAB has a reduced cold-start period in the early stages of operation as users will receive two sources of recommendations. Unseen items can immediately be recommended to users, while the collaborative aspect allows recommendations to be

formed based on human perception and an not simply the ‘dry’ machine processing algorithms. Users will rates pages based on their presentation and comprehension as well as their content thereby reducing badly composed or presented documents. Together these features, in addition to the underlying collection/selection model, help FAB to “scale gracefully as the number of users and documents rise” (Balabanović & Shoham, 1997).

5.4 Summary

Due to their frequent reliance on agent technology and the close similarity with the field of adaptive hypermedia, this chapter has provided an overview of the work undertaken by the recommender system community. This includes a look at the types of recommender systems that exist, how they relate to adaptive hypermedia techniques and practices, and also providing several examples of work in this field.

Having covered the relevant research fields, topics, issues and history behind the work documented in this thesis, the next chapter will describe the early steps towards developing an agent-based framework for adaptive hypermedia. These initial systems extended and combined some of the ideas from the adaptive hypermedia, open hypermedia, recommender system and agent research areas.

Chapter 6

Initial Work

6.1 Introduction

This chapter presents the initial investigation into adaptive hypermedia by the author. The influences of agents, SoFAR, open hypermedia, Microcosm and link services led to the development of the first agent-based adaptive hypermedia system, termed the Personal Agent information And Delivery System (PAADS). This system demonstrated the possibility of using an agent-based system for adaptive hypermedia, and also made use of open hypermedia linkbases as a means for providing the adaptation.

Following this, a second system, Augmented Linking In Context (ALIC), was developed that enhanced and extended many of the ideas present in PAADS. This second approach used the notion of a user's spatial context within a hypermedia domain as a means of profiling the user, and then a link server matched this model against a set of linkbases to provide augmented links to the user.

6.2 The Microcosm Legacy

The early open hypermedia system Microcosm (Section 3.2.3) was originally designed as a test-bed for hypertext research and it has been very successful in this regard. Microcosm spurred on many advances in hypertext and open hypermedia research and helped raise the importance of the basic navigational link. The development of systems such as the DLS (Section 3.2.4), MEMOIR (Section 5.3.1), QuIC (Section 5.3.2) and the recent Xlink standard (W3C, 2002) have all been heavily influenced by results of the Microcosm project. It is also possible to look back on the

innovative features of Microcosm in the light of more recent technological advances. Two such reflections have directed much of the early work presented in this chapter.

6.2.1 Microcosm as an "Agent-Based" System

The advancement of technology in the years since Microcosm's development, in particular the emergence of agent technology, has highlighted many of Microcosm's features as being ahead of their time. The modular nature of the Microcosm system and the design process that included configurability as a primary concern, are both aspects that would probably cause a similar system today to be implemented as a multi-agent system. Indeed, the modular nature of Microcosm is also apparent in a more recent attempt at distributing the core components of the Microcosm system (Goose *et al.*, 2000).

With hindsight, it is clear that Microcosm's filter chain mechanism closely resembles the communication process employed by modern software agents. Filters in Microcosm had a dynamic property allowing them to be introduced, removed or re-organised within the chain in real time by the user. Filters were registered with a central broker called the filter manager, which recorded the state of each filter and its position within the chain. A similar mechanism can be found in almost all agent frameworks, which allow them to start, stop, and if possible migrate at will without causing disruption to the remaining agents. This is due to the communication architecture used in agent-based systems. Most agents employ a central registry service (or broker agent), which other agents can use to ascertain the existence and location of a receiving agent before establishing a conversation.

The Microcosm filter chain mechanism provided the first indication that agent technology could be applicable to hypermedia systems. It also highlighted the fact that the specific agent technology underpinning the system was not as important as the intrinsic properties of modularity, configurability and flexibility embodied by every agent.

6.2.2 Microcosm as an Adaptive Hypermedia System

Microcosm was primarily aimed at solving the problems associated with large hypertext systems of its time, problems then being tacked by open hypermedia systems, however its flexibility and heterogeneous approach to hypermedia design meant that Microcosm was applicable to a wide variety of application areas. This was aided by the filter chain mechanism, which allowed new functionality to easily be introduced into the system. One use of these filters could be to keep a model of user actions and then provide personalised navigational links based on this user model. Microcosm was first used as an adaptive hypermedia system for an experimental analysis of various adaptive techniques (Hothi & Hall, 1998). The system assigned a stereotyped static user model based on the results of a pre-questionnaire. Stereotyping (Rich, 1979) is a technique used to categorise a user, often into novice, intermediate or expert classes, and create a user model with a set of default values for each of these classes. This provides the system with a foundation from which to start providing adaptation, thereby overcoming the initialisation problem. Using a set of linkbases, Microcosm provided adaptive navigation by applying a different linkbase for each stereotype model. Adaptive presentation was achieved by considering a user's skill level and then applying a shading effect to the text to indicate denser expert material.

This perspective of Microcosm as an Adaptive Hypermedia System (AHS) caused the team to rethink the impact that open hypermedia could have on the field of adaptive hypermedia. Techniques that fall under the umbrella of adaptive navigation support are essentially concerned with manipulating hyperlinks. Microcosm and the DLS demonstrated the importance of linkbases and link services for creating and maintaining sets of hyperlinks. Therefore, a technology such as this should be equally applicable to systems employing adaptive navigation support techniques. These concepts influenced the development of the first AHS developed for this research.

6.3 PAADS

The initial aim of the Personal Agent information And Delivery System (PAADS) (Bailey & Hall, 2000) was to examine the use of linkbase technology, developed from the open hypermedia community, as a means of enabling adaptive navigation support techniques. A second, but equally important aim, of this early

system was to explore the use of agent technology as a foundation for implementing AH. The result of this work produced the first reported agent-based adaptive hypermedia system using a link server to provide navigational support; specifically adaptive augmented linking.

SoFAR (see Section 4.6.3) was the agent framework chosen to implement PAADS. This decision was taken, partly due to the generic nature of most agent frameworks as they essentially offer all application authors the same basic functionality, and partly due to the support available to SoFAR users within the IAM research group. As demonstrated by Microcosm, QuIC and most collaborative recommender systems, agents are ideally suited to dynamic situations where information needs to be gathered from a variety of sources and then manipulated by different entities, while certain data will persist over time, such as a user's interests.

6.3.1 The PAADS System Architecture

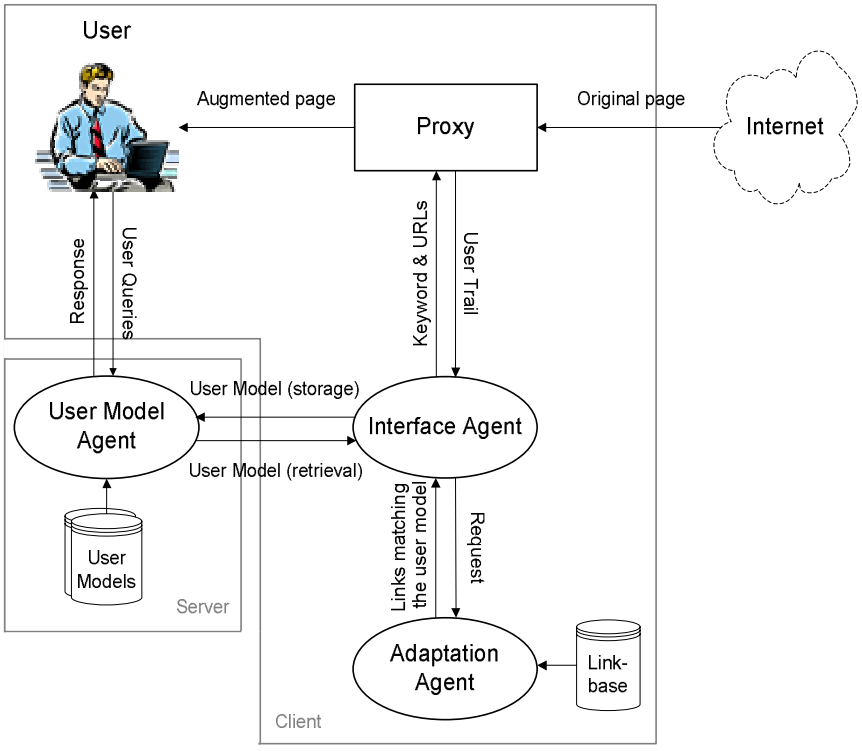

Figure 6-1. The PAADS system architecture

The PAADS architecture, Figure 6-1, is designed as a client-side proxy-based AHS with a centralised storage area for user models, allowing them to be shared. It is

termed client-side because the adaptation mechanisms reside on the user's machine. User adaptation occurs through a local proxy which sends messages on to an interface agent. The user's proxy records the title of each page visited by the user and passes this information onto the interface agent, which identifies frequently occurring words in the titles of each visited page. These keywords are stored as the *interests* field in a user model, maintained by a separate user model agent. The user model agent provides the user with complete access to their user model via a set of servlets; small programs which respond to HyperText Transfer Protocol (HTTP) requests from a web browser. Before gaining access to the information in their profile, the user must authenticate themselves with the servlets using a username and password combination. After this, they are presented with their profile, as shown in the screenshot in Figure 6-2.

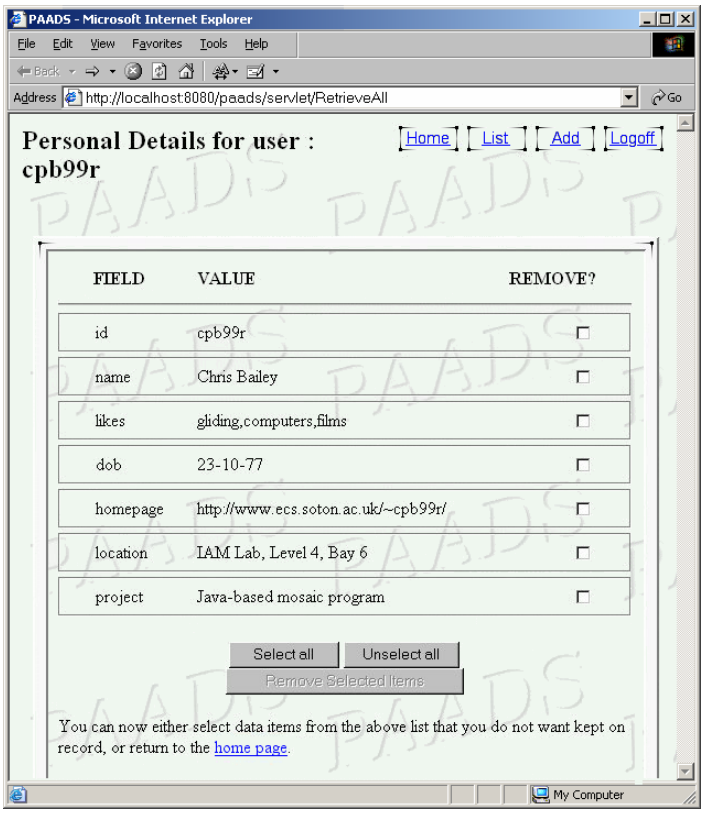

Figure 6-2. PAADS screenshot of a user accessing their profile

The profile consists of entries such as name, date of birth, location, homepage URL and the automatically generated interests field, which contains the current set of identified keywords. Aside from the interests field, all the other information is entered explicitly by the user, usually on their first interaction with the system. Users can add

new information, edit existing entries and remove old information. Users are also allowed to edit the interests field built up by the interface agent.

PAADS is intended to exist within an academic community of users. The original scenario envisaged for PAADS presented a situation where each user of a community would be assigned a personal agent to trail them and slowly build up a profile of their status within the community. In this situation, the personal user agent would track the various projects, reports, documents and meetings the user became involved with. Information mined from these and other sources such as home pages would contribute to a user model. These user models could then be pooled together providing a resource for anyone to query.

This scenario resulted in the design of the user model agent as a central repository of user models residing on a separate server. This agent provides a query mechanism, again using servlets, allowing a user to query the user model agent and obtain information relating to members of the community. Queries such as "Who is *X*", "Where is *X*", "Who can help me with *X*" and "I'd like to find someone knowledgeable about *X*" can be composed by the user using a web form, and sent to the agent. For each type of question there is a matching template (see Figure 6-3) that instructs the user model agent on how to handle the query. For example, the "Who is *X*" query searches the id, name and email fields for *X* and returns the id, name, email, location and homepage information, if available.

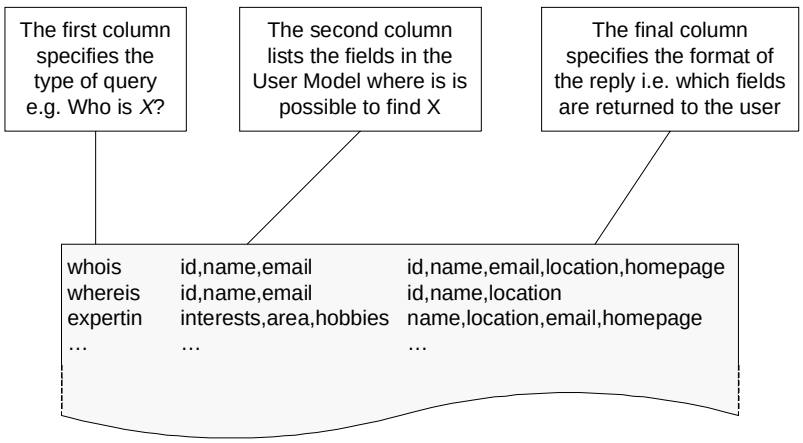

Figure 6-3. An example of the PAADS query template file

The adaptation engine in PAADS contains a single linkbase, constructed by hand. Each link consists of a URL and a set of associated keywords that describe the link. When the interface agent queries the adaptation agent, it sends the keywords identified from the titles of visited web pages. The adaptation agent matches these keywords with the keywords from each link in the linkbase and returns the matching subset of links. The links are then sent to the proxy, whose job is to insert the URLs into new documents whenever an associated keyword is identified in the body of the web page.

The PAADS system demonstrates that it is indeed possible to construct adaptive hypermedia applications using a set of agents. This work presents several novel concepts to the adaptive community.

1. *Developing agent-based adaptive hypermedia applications*. PAADS provided one of the first concrete examples of how an agent community could be used to provide adaptive hypermedia. To support this, the agents used in the development of PAADS adhere to a stricter definition of agenthood than that of rudimentary communicating processes.
2. *The use of link services for adaptive hypermedia systems*. Prior to PAADS, link services for adaptive hypermedia were only to be found in WBI (Maglio & Farrell, 2000) and Microcosm, both of which have been reported outside the field of adaptive hypermedia and which have had adaptive features bolted onto their existing functionality. In contrast PAADS demonstrates the first use of a link service primarily as a means to provide adaptive navigation support.

Although PAADS took a simple approach to many aspects of the system, the successful interaction of link services and a multi-agent system prompted the design and development of the second AHS. In many ways, this next system can be seen as the second generation of PAADS technology even though little of the PAADS code exists in the second system. The particular aspects of PAADS that have been extended in the next generation system, ALIC, include removing the reliance on a proxy to deliver the augmented links, the use of multiple linkbases by the link service and the consideration of a user's context when providing adaptive navigation support.

6.4 Augmented Linking in Context

The results of designing, building and documenting PAADS provided a stimulus to continue developing the concepts employed in the system. Augmented Linking in Context (ALIC) is the second AHS developed for this work. It incorporates the context approach developed in the earlier system QuIC (Section 5.3.2), with the link service agent from PAADS to build an AHS that provides context-based augmented linking (Bailey *et al.*, 2001).

The original concept for this system can best be understood by using the analogy that browsing the web is a lot like watching a movie. A movie consists of a linear series of frames, and a sequence of these frames constitutes a scene. Many scenes make up the whole movie. In this way it can be stated that:

A movie consists of a set of scenes, which are composed of a sequence of frames.

If this movie model were compared to a user's trail as they browse the web using a single browser window, it can be said that a user's trail is then made up of a linear series of pages. As a user continues to browse over a length of time, they will inevitable change their topic of interest. Therefore:

A trail consists of a set of topics, which are composed of a sequence of URL's.

These topics arise because the most common form of page transition is the hyperlink. Hyperlinks denote the locations of related information, therefore two URL's one after another in a trail have a high chance of being closely related. When a user trail is viewed as a series of topics, the topic borders can be either *hard*, as in an instantaneous scene change, i.e. the user might jump to a document by selecting a URL in their favourites list, or *soft*, where the topic changes gradually through a series of related URLs. Soft topic boundaries have a similar analogy to the fade device used in movie scenes. The theory behind ALIC stated that if an application could detect these scene changes in the user's browsing experience, then it would be able to apply a specific linkbase for the current topic. The technique chosen to identify topics in the users trail is the context algorithm developed for the QuIC system (described in section 5.3.2).

6.4.1 The ALIC System Architecture

From a system perspective, the major difference between the two systems PAADS and ALIC, is the adoption of a hybrid system approach in the latter that uses features from both client-side and server-side programming paradigms. Table 6-1 lists the most important differences between the two systems. The reason for using a hybrid approach was that the proxy used in PAADS was seen as a major inhibiting factor on the range of possible adaptive features that could be provided. Rather than rely on a user to install, configure and run a proxy as PAADS required, a new approach was taken where a smaller application runs on the user's machine and seamlessly hooks into the user's browser. Using the Object Linking and Embedding (OLE) features of the Microsoft Windows operating system that provides a mechanism for cross-application control, this program allows the browser to operate independently while keeping track of the user and providing both adaptive content and presentation. Like PAADS before it SoFAR agents are used to implement the server aspects of ALIC. The decision to use agents is primarily based on the success of the agents in the PAADS system.

SYSTEM	ARCHITECTURE	METHOD OF USER INTERACTION	BASIS FOR USER MODEL	ADAPTIVE TECHNIQUES SUPPORTED
PAADS	Client-Side	Web Proxy	Keywords extracted from the titles of web pages	Augmented Linking
ALIC	Hybrid	Microsoft Internet Explorer OLE controls	Clusters of TFIDF vectors from previously visited web pages	Contextual Augmented Linking

Table 6-1. A table comparing the two developed systems, PAADS and ALIC

The three agent roles used in PAADS have been replicated in ALIC. The similarity between new and old systems also meant both the link service agent and user modeller agent required little alteration. The interface agent in ALIC required a complete rewrite to reflect the new method of user interaction. The interaction of these agents is shown seen in Figure 6-4. Together the three agents attempt to identify a user's context from the pages they visit, and then match this context up to a specific linkbase. This linkbase becomes 'activated' whereby its links are augmented into all new web pages, providing the user stays within the same context. When the user changes context, i.e. browse for documents relating to a different context, the system re-evaluates the context and finds a new linkbase which is then activated. Another difference between the ALIC system and the PAADS system, is that PAADS contained

a single linkbase and activated the *links* that closely matched the user's model. The new approach uses several smaller linkbases, each containing links pertaining to a specific topic. The current context is matched against each linkbase in turn and the closest matching *linkbase* becomes activated. The page is then augmented with each link from the linkbase. By activating an entire linkbase rather than simply the matching links, a user receives additional links related to, but that not necessarily match, their context which helps broaden the range of links they receive.

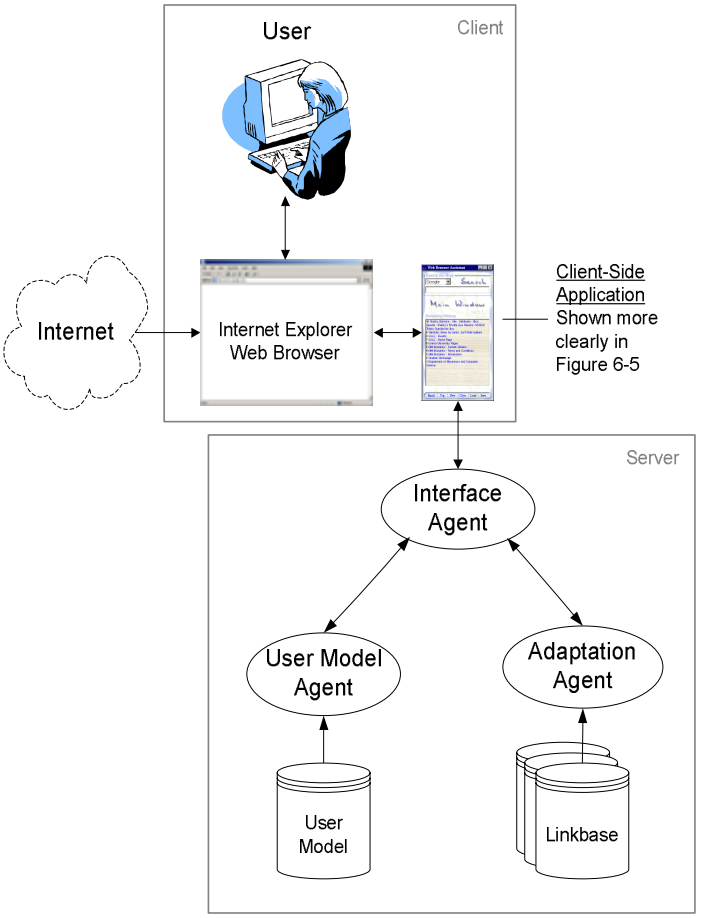

Figure 6-4. The ALIC system architecture

The approach in ALIC was to use the TFIDF document vector algorithm, as used in the QuIC system, as a means of dynamically building clusters of document vectors that represent the user's context as they move through the web. To do this, the client-side application tracks the user, recording which pages they have viewed and sending this information onto the interface agent. The interface agent downloads the web page for the associated URL and calculates the document vector for the page using the QuIC context algorithm. The user model agent receives this document vector from the

interface agent and assigns it to a cluster. It does this by computing the similarity between the current document vector and each existing cluster with a cosine similarity function. The document vector is then added to the cluster with the highest match. If no clusters match the vector, or if the highest cluster match is below a given threshold value, then a new cluster is created with the initial values obtained from the document vector and the system assumes the user has moved onto a new context. Once a new context is identified, the user model agent informs the interface agent which in turn composes a request to the adaptation agent. The adaptation agent maintains a set of linkbases related to various subjects. At the request of the interface agent, the adaptation agent compares the new cluster formed by the user model agent with each linkbase. Keywords from the cluster are matched against the keywords for each link in each linkbase. The linkbase with the highest number of matches above a certain threshold becomes activated and its links are returned to the interface agent who in turn passes them onto the client-side application, shown in Figure 6-5.

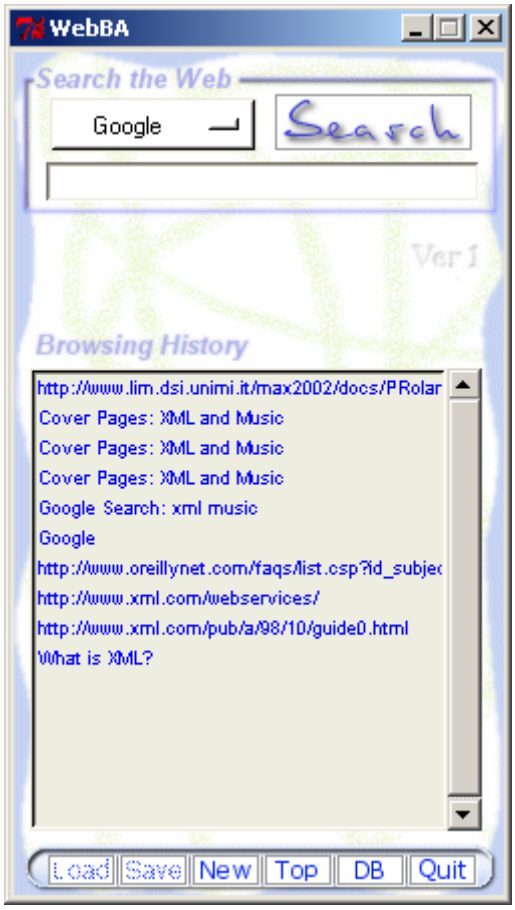

Figure 6-5. Screenshot of the ALIC client-side application

The application runs on the client machine and forms a handle to the user's web browser using an OLE object provided by the browser application. This object provides a set of properties and methods for controlling a third party application. OLE objects can be accessed using most programming languages. The Internet Explorer browser object provides methods for loading new pages, determining the location of the current page, and methods to read and write information in the current browser window. After a new user context has been identified and the client application receives the new linkbase, it extracts the contents of the currently displayed window via the OLE object, and analyses the text for linkbase keywords. These keywords are then converted into URLs and the final page is then sent back to the browser window, overwriting any existing information being displayed. The user sees the original page load in the browser and then the text is overlaid with the new links. The only change that occurs in this process is when the server detects the user changing context and then the client application will receive a new set of contextual links from the adaptation agent. Figure 6-6 demonstrates how a page would appear to a user if they visit the page from two different contexts.

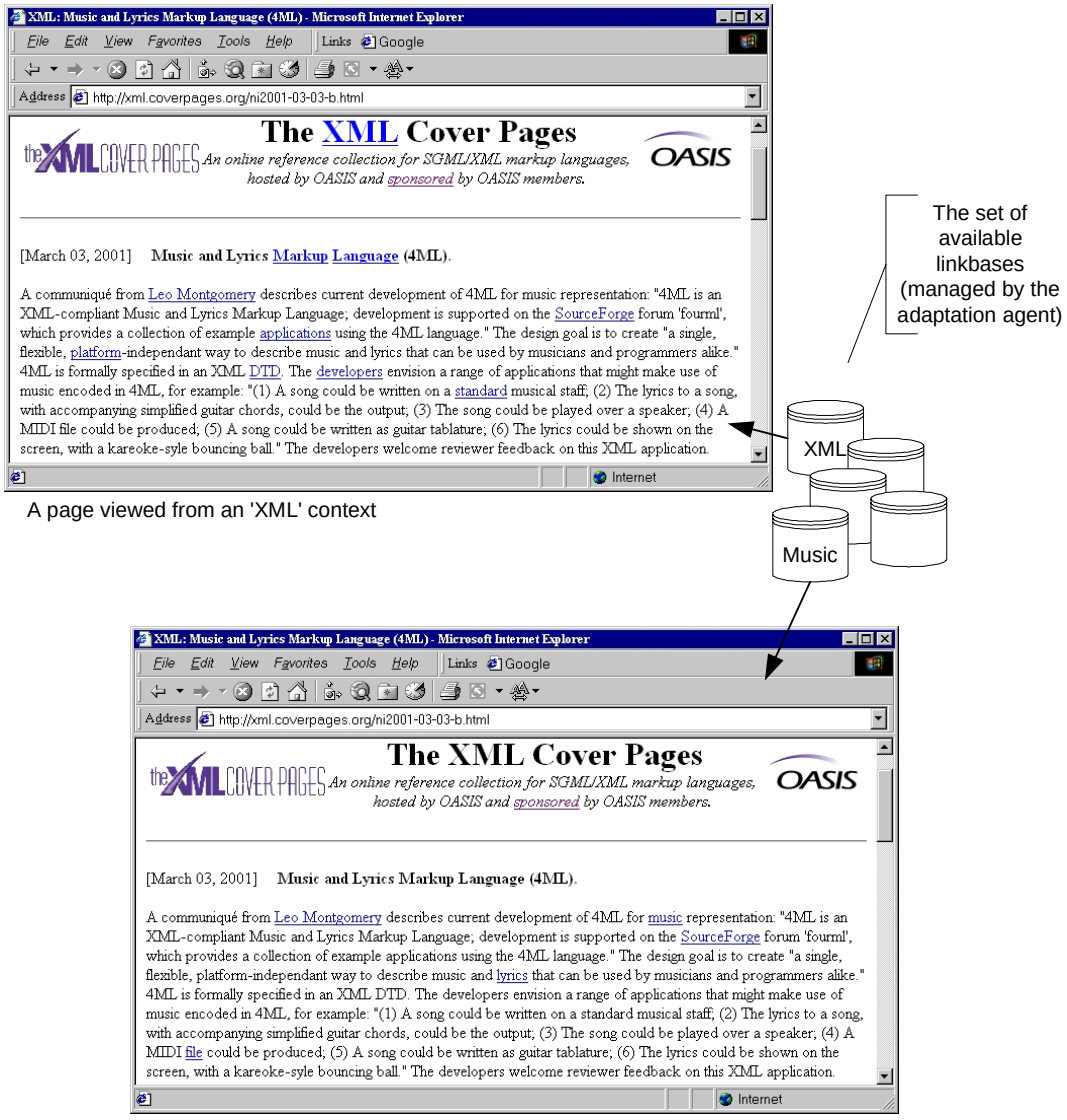

Figure 6-6. Example of how a page appears when a different linkbases are activated

This method of augmenting the page with contextual links is only as successful as the quality of the links and linkbases managed by the adaptation engine. As with PAADS, each linkbase was carefully constructed manually to contain links pertaining to web sites with useful information about a particular subject. This reliance on the quality of links is a major concern to any adaptive hypermedia system employing linkbases as a source of adaptation. If this system were to be extended, an obvious area of future work would be the generation of linkbases using ‘intelligent’ resource gathering agents or more simple web crawlers; software algorithms that autonomously analyse large numbers of web pages. Such techniques are already being used, for example in Portal Maximizer (Active Navigation, 2002), the commercial product that evolved from the DLS (Section 3.2.4).

6.5 Reflections on PAADS and ALIC

Experiences with PAADS and ALIC produced a wealth of ideas for the future direction of this work. While PAADS was the first agent-based adaptive hypermedia system incorporating a link service, ALIC extended these concepts, taking context algorithms developed for QuIC and applying them to the domain of adaptive hypermedia.

The influence of open hypermedia concepts, specifically linking technology, on both systems indicates the potential for crossover between the fields of open hypermedia and adaptive hypermedia. Linkbase technology is only one aspect of open hypermedia which can be applied to adaptive hypermedia. Chapter 8 will continue to explore the relationship between the two fields by introducing the more recent work on open hypermedia models into adaptive hypermedia.

One of the primary similarities between the two systems is that they are both built using agent technology. Moreover, the structure and functions of many of the agents are very similar in nature. These agents have been used to implement two different architectural paradigms. In PAADS, a server-side approach has been taken, while ALIC opts for a hybrid approach, which combines aspects from classical server-side and client-side methodologies. This combination of approaches demonstrates the diversity that using an agent architecture can bring to programmers and system developers.

6.6 Summary

This chapter has followed the development of two adaptive hypermedia systems, PAADS and ALIC. These systems have many novel and interesting features that have been detailed, however the chapter primarily provides the foundation towards developing an agent-based framework to support adaptive hypermedia and adaptive web-based systems. These systems demonstrate the application of agents in building AHS. Agents provide rapid application development within a reliable environment and aid future development by providing self-contained reusable components.

The next chapter presents a novel agent-based framework which can be used to implement a wide range of adaptive applications (both hypermedia and web-based). This framework is geared towards providing a middle ground between the formal AHAM model and the other implementation-oriented frameworks.

Chapter 7

ABAH: An Agent-Based Framework for Adaptive Hypermedia

7.1 Introduction

This chapter presents a new Agent-Based framework for Adaptive Hypermedia (ABAH), which attempts to fill a gap between existing formal models that formally specify a system, and implementation-specific frameworks used to build real applications. The motivation and justification for developing a new framework will be presented. Firstly there is an analysis of core components present in existing adaptive hypermedia systems and this is followed by the development of a set of agents, based on these components. The agents and associated knowledge models together form the framework. This chapter concludes by presenting some of the advantages of the ABAH framework.

7.2 The Case for a New Framework

All the Adaptive Hypermedia (AH) frameworks described in Chapter 2 can be said to fall into one of two categories; formal models or implementation-specific frameworks. The formal models, AHAM, Munich and XAHM are used to express the functionality of new and existing Adaptive Hypermedia Systems (AHS), while the frameworks ELM-ART, ACE and MetaLinks are aimed at implementing a specific type of adaptive application. There is an absence of middle ground between the formal AHAM and the implementation-specific frameworks. A formal model can describe a system but has no means of implementing it, while an implementation-specific framework can be used to build a type of physical system but lacks true generalisation and a formal mechanism for conveying the properties of that system. Currently the

closest approach is that taken in the Munich reference model, but it focuses on guiding the designer through the lifecycle and not on supporting the application programmer with a variety of approaches.

Another issue is the increasing popularity of adaptive *web-based* systems that are starting to distinguish themselves from existing adaptive *hypermedia* systems. While these terms are often used interchangeably, this thesis draws a distinction between the more traditional adaptive hypermedia applications, which provide adaptation to static hypermedia web pages, and the newer service-orientated adaptive web-based systems which offer personalised or tailored services to dynamic data sources. Examples of adaptive web-based systems can be found in personalised newspapers (Sunderam, 2002) or TV guides (Baudisch & Brueckner, 2002).

The ABAH framework that is presented in this chapter will define a loose architecture around which many types of AH applications can be created and which can also be used as a vehicle to implement formal models for AH. The framework is designed from a systems perspective, with the aim of producing an architecture that is flexible enough to be used to implement the wide variety of existing adaptive applications, both hypermedia and web-based. The framework also specifies the types of knowledge models used by AHS, strengthening the link from the formal methodology used in AHAM, to the framework and then final system implementation. The objective is that this framework, which focuses on modelling each component of an AHS, can form a bridge between the two extremes found in existing framework design; a middle ground between formal modelling and functional implementation.

7.2.1 Why Develop an Agent-Based Approach?

Agent technology is largely ignored within the AH community. Mention of agents within the literature is very sparse, for examples see (O'Hare *et al.*, 2000)(Silveira & Vicari, 2002) and often used in combination with a recommendation service (Pazzani & Billsus, 1999)(Lieberman, 1995). It is also hard to gauge how closely these 'agents' adhere to the more formal definitions of agenthood as given in Section 4.3. Well-defined agent-based systems have been largely ignored in spite of the wider uptake in related fields such as intelligent tutoring systems (Giraffa & Viccari,

1998) and recommender systems (El-Beltagy *et al.*, 2000)(Balabanović & Shoham, 1997).

Agents provide a range of advantages over conventional programming methods, some of which were discussed in Chapter 4. However one particular advantage from the perspective of building a framework, is the flexibility that agents provide. The ability to run on multiple platforms, across different environments and with the facility to move between hosts provides a versatility and freedom to agent-based systems that requires significant effort to implement through conventional programming. This flexibility also appears in the two systems, PAADS and ALIC. They both take different architectural approaches in terms of the location of each agent and the mechanism for delivering the adaptation. This versatility is critical for developing a framework expressive enough to describe the range of AHS, especially if it is to be positioned between formal models and implementation-based frameworks.

7.2.2 Types of Existing Adaptive Hypermedia Systems

Currently, AHS can be classified into one of three different classes. This categorisation is based on the physical location of components within AHS, which in turn causes restrictions to be placed on the range of adaptive technologies that can be used. After analysing the existing approaches to adaptive hypermedia and adaptive web-based systems in terms of adaptive features offered and the underlying physical implementation used, three types of systems can be identified; client-side, server-side and hybrid systems. The main distinction concerns the location of the adaptation component within the architecture. These system classifications are not restricted to adaptive applications but can also appear in non-adaptive applications, where a wider range of system types can be found, including peer-to-peer and distributed applications. Appendix A provides a table of various adaptive hypermedia systems classified by the location of their adaptation engine.

Server-Side Systems

LOCATION OF ADAPTATION COMPONENTS	ADAPTATION APPLIED TO	AH TECHNIQUES USED	ADVANTAGES
Web site server	A specific web site / domain	All forms of adaptive presentation and adaptive navigation support	High level of domain understanding Processing power No installation costs for user

Table 7-1. Features of server-side adaptive hypermedia systems

Table 7-1 presents server-side systems; the classic form of adaptive web sites which have their foundations in intelligent tutoring systems. These systems usually provide an adaptive module situated alongside or included in a web server. User modelling information is obtained via explicit questioning, tests or implicitly from a web log. The server-side AHS uses this information to construct a document from either the raw domain material or existing web pages and passes the page onto the web server which returns it back to the client. Server-side systems usually provide adaptation across an entire information domain and these web sites often support both adaptive presentation and adaptive navigation support. Because of the close proximity between the application and the domain information, server-side systems can gain a greater level of domain understanding, which allows them to provide more highly detailed and precise forms of adaptation. It is because of these advantages that most of the successful adaptive hypermedia systems such as INTERBOOK (Section 2.6.1) and AHA! (Section 2.6.2) provide such good examples of server-side adaptation.

Client-Side Systems

LOCATION OF ADAPTATION COMPONENTS	ADAPTATION APPLIED TO	AH TECHNIQUES USED	ADVANTAGES
Client (user) machines	Potentially every visited web page	Primarily adaptive navigation support	By being located closer to the user, client-side system can more accurately model the user

Table 7-2. Features of client-side adaptive hypermedia systems

Client systems, see Table 7-2, run the adaptation mechanisms on the client's machine where the close proximity with users allows for a wide range of user features to be obtained. Collecting and maintaining user profile information solely on the client's machine also provides greater user security, an important issue among AH systems (Bradley et al., 2000). The approach of situating the adaptive application on the client's machine allows closer access to the user, and the possibility to provide adaptation to any web page the user visits. The disadvantage of such an approach is that in moving away from the server, client-side AH systems have little or no structured domain knowledge, which restricts the type of adaptation they can provide. Amongst most client-side adaptive applications, the most common form of adaptation is navigational support techniques based on web page analysis, often using information retrieval and machine learning algorithms.

There are various implementation issues associated with client-side programming which hinders their development. The most crucial aspect is the need to support as many users as possible, which necessitates supporting multiple platforms, multiple browsers, different browser versions and users with several computers. The task of keeping up with browser types and versions is incredibly time consuming for developers, and often necessitates the use of an automatic application update routine as it is not always possible to rely on the user alone to keep the system up to date. Other important issues when deploying client-side applications include the ease of installation and the amount of configuration necessary after installation; overheads that place unnecessary pressure on users and can affect their choice of application. Examples of client-side AHS include PAADS and Letizia (Lieberman, 1995).

Hybrid Systems

LOCATION OF ADAPTATION COMPONENTS	ADAPTATION APPLIED TO	AH TECHNIQUES USED	ADVANTAGES
Independent server (sometimes in addition to a small client application)	Potentially every visited web page	Adaptive navigation support	Processing power, low installation costs and better user modelling

Table 7-3. Features of hybrid adaptive hypermedia systems

A small set of AHS have attempted to bridge the gap between client and server-side approaches, referred to here as hybrid systems, see Table 7-3. They do not reside on either the user's machine or the web server. Instead they are hosted by separate machines, independent of either the client or the web server. Hybrid systems provide adaptation in a similar manner to client-side systems, while overcoming the implementation issues associated with them. To complement this, the additional processing power available to servers allows hybrid systems to include a range of resources and domain knowledge to enhance the quality of the adaptation provided to the user. In spite of their advantageous features, hybrid systems are relatively rare. Examples include WBI (Maglio & Farrell, 2000) and ALIC.

When developing a generalised framework to implement any adaptive hypermedia system, it needs to be expressive enough to cope with each of these three system designs. This problem can be tackled by using an agent architecture that supports mobility among its agents. Extending this concept further and acknowledging the ability of agents to re-configure themselves to best solve a problem, it should be possible to implement an agent system that shifts between client-side and server-side states depending on the current requirements of the user and the resources available to each agent.

7.2.3 Issues with Existing Frameworks

Having presented the frameworks for adaptive hypermedia in Section 2.7, there now follows a critical analysis of the approaches they each take.

The Oxford dictionary (Smart, 1997) defines the word *framework* as follows:

> *framework* : **1** essential supporting structure.
> **2** basic system.

It is clear that each framework presented in Section 2.7 attempts to provide a supporting structure for building AHS. In the cases of ELM-ART, ACE and MetaLinks, the support these frameworks provide is centred around a single application or application domain. ELM-ART and ACE provide frameworks for adaptive courseware (educational systems) while MetaLinks is primarily aimed at adaptive

hyperbook designers. The difference between these two approaches is that courseware systems use some form of online skills testing, such as multiple choice questions or programming problems as an aid to user modelling. In contrast to all three approaches, the high-level formal nature of the Adaptive Hypermedia Application Model (AHAM) and XML Adaptive Hypermedia Model (XAHM) do not refer to any particular domain. As a result, if a system has been specified in AHAM it can easily be compared to other AHAM-specified systems but it is no closer to actually being implemented except that now programmers have a clear idea of how each component operates and interacts.

Another issue concerns the existing set of implementation-oriented frameworks that are too closely tied to their application domains. This results in an additional effort needed to re-implement systems when transferring to a new domain. Systems that offer layers of abstraction on top of the raw information in the domain provide flexibility not just to the application area, but also to the adaptive techniques and architectural implementation used. Abstraction also allows far greater interoperability between systems, provided a common language has been formally agreed upon. This problem of system interoperability has been examined in Section 2.8.3.

One of the advantages of a generic framework is its ability to express and implement a wide range of adaptive hypermedia and adaptive web-based systems. This includes the architectural designs described in Section 7.2.2. Of all the systems discussed, the nearest contender to this idealised framework would be AHAM. Unlike XHAM which enforces certain restrictions on system designers, such as the use of stereotyping, AHAM does not define any implementation specifics within the formal model thereby providing programmers with the freedom to use any number of methods they like to implement the system, providing it conforms to the AHAM-expressed version. However as yet there are no tools or techniques designed to aid AHAM system developers.

7.3 Developing the Framework

The first stage in developing ABAH involved identifying the major components of existing AH systems, models and frameworks. While developing the two earlier systems, PAADS and ALIC, one of the interesting similarities that was found between these two approaches, is that each agent in PAADS matches an ALIC agent that

provides the same basic functionality. This finding occurs in spite of the fact that each system uses a different form of adaptive link augmentation and a different architecture arrangement. When compared to components in other systems these similarities became more obvious. These components are also clearly highlighted in the frameworks for AH. Using the experience gained from PAADS, ALIC, existing frameworks, and the plethora of AHS in existence, the following table has been formed which lists the primary components required for all AHS investigated, in addition to the most common storage models employed by these components.

STORAGE MODELS	SYSTEM COMPONENTS
Domain Structure	
Domain Data	User Modeller
Adaptation Rules	Adaptation Engine
Pedagogical Model	Interface Component
User Model	

Table 7-4. Key components of adaptive hypermedia systems

The storage models listed in Table 7-4 have already been described in Section 2.8.1. However the system components require further explanation as they each encompass many features of AH and might appear under different names and guises in other systems. In most adaptive hypermedia systems, three components work together to provide adaptation; the *user modeller*, *adaptation engine* and *interface component*. It should be noted that in some systems these components are not always implemented as individual units, instead programmers might opt to combine them into larger units or break them down into smaller ones. For example, the interface component can be split into a graphics generating part and a page content generating part as implemented in the PUSH system (Section 2.6.3).

7.3.1 User Modeller

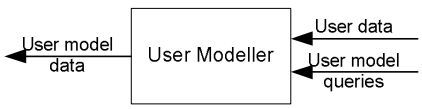

Figure 7-1. The user modeller

The role of the user modeller, Figure 7-1, is to store, recall and process user profile information. It is not the function of the modeller to obtain user data as this task is assigned to the interface component allowing it to present a unified and consistent user interface. New data is sent to the user modeller by other system components and it responds to queries concerning information in the user model. Although security is not examined in this research, the user modeller component of a real system would have the responsibility to protect and maintain user privacy, and provide authentication mechanisms. An example of one such design for a user model server is demonstrated in the Personis system (Kay *et al.*, 2002).

7.3.2 Adaptation Engine

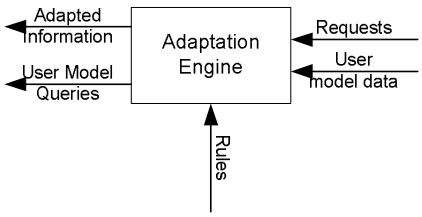

Figure 7-2. The adaptation engine

The Adaptation Engine (AE) shown in Figure 7-2, closely resembles the AE outlined in AHAM. With access to domain concept data and the associated structure, the AE responds to requests from the user or other system components and applies adaptation rules to the domain to produce adaptive information, most often in the form of web pages. These rules may be stored in databases, sometimes referred to as rulebases (Brusilovsky, 1996a), or hard-coded into the application. The AE also requests user information from the modeller and uses this as input into the adaptation rules. It is possible that activation of these rules triggers a user model update and in such a case the AE would be responsible for informing the modeller when new information becomes available.

7.3.3 Interface Component

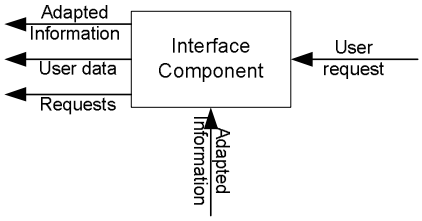

Figure 7-3. The interface component

The role of the interface component, Figure 7-3, can be very complex but it has two primary roles. Firstly, it is responsible for every aspect of the user interface. This includes the style of the pages, whatever installation and setup are necessary, tracking the user and obtaining user preferences, either explicitly or implicitly. The second aspect of the interface component is responding to user requests, sending profile data to the modeller, formulating queries to the AE and presenting the results to the user. The interface component acts as the 'glue' between the user modeller and adaptation engine. It is responsible for informing the modeller of new profile information and initiates adaptation requests to the AE. Together all three components form the central body of an AHS.

These components form an arrangement as shown in Figure 7-4.

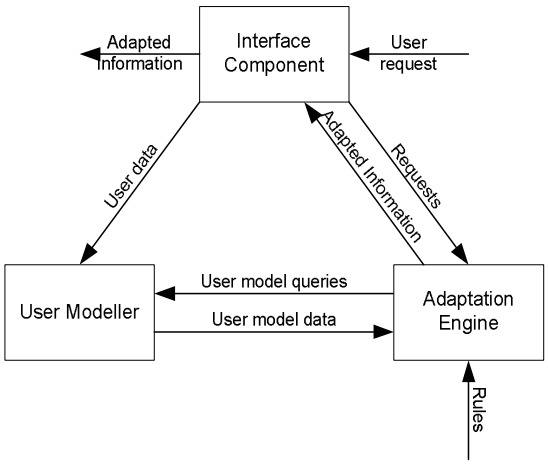

Figure 7-4. Components of an adaptive hypermedia system

7.4 The ABAH Framework

After identifying the basic components of all adaptive hypermedia systems, it is possible to assign an agent to each component. With the roles described above, the arrangement of agents results in a framework that contains all the basic functionality present in any AHS. This Agent-Based framework for Adaptive Hypermedia (ABAH), pronounced *"Abba"*, is shown in Figure 7-5 and consists of a user model agent, adaptation agent and interface agent, and details all the possible communication channels between these three agents. The diagram also shows the possible interaction methods between agents and the data models needed by each agent.

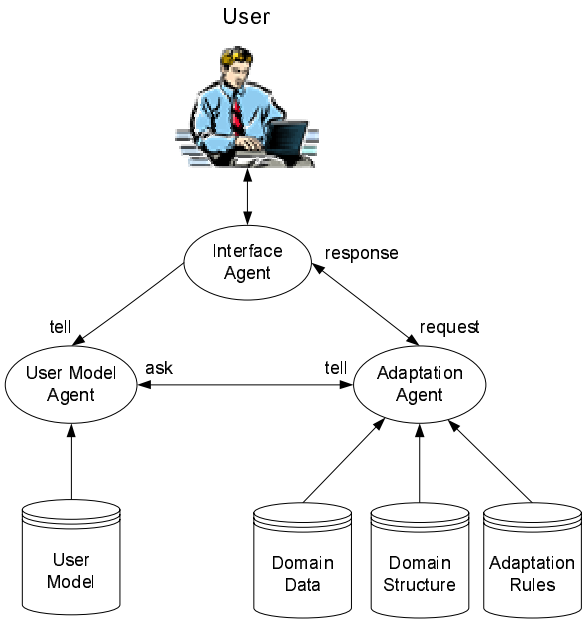

Figure 7-5. The basic ABAH framework

Due to the conceptual nature of this framework, no specific infrastructure is defined, nor a particular data format imposed onto any of the data storage models. The only properties that are required by ABAH are the basic communication and resource discovery features available in all agent frameworks. Although it is unnecessary to formally specify the semantics of a communication mechanism, the language must support basic message passing performatives to allow agents to request information (`ask`), inform other agents of new information (`tell`) and subscribe to agent events (`subscribe`).

The second condition imposed on the underlying agent infrastructure is the ability to advertise and discover new resources. Agents need to be able to obtain new information and the environment must provide the opportunity for new agents to advertise the resources they offer. Resource discovery also encourages competition between agents offering the same services. This leads to service negotiation and bidding scenarios, features that can ultimately benefit end users.

7.4.1 User Model Agent

The user model agent assumes the role of the user modeller component. All user profile details are processed by this agent. Essentially a reactive agent, it receives new information through `tell` performatives from the interface agent, and responds to `ask` requests from the adaptation agent. In an agent environment, other agents should be able subscribe to the user model agent and receive updates whenever new user information becomes available.

7.4.2 Adaptation Agent

The adaptation agent provides the processing required to adapt information to a user's requirements. The adaptation agent resides over three information models. The information domain is split into two models; the unstructured information is contained in the domain data file, and the domain structure (concepts, links, relationships) in a separate file. Storing data models separately using a standard representation format is a concept open hypermedia systems use to facilitate data reuse and increase interoperation between systems (Wiil & Whitehead, 1997). It is entirely possible to combine these two models together. The third model, the rulebase, contains the rules for applying adaptation to a particular domain. These rules might require user model input, obtained via requests to the user model agent.

The adaptation agent assumes the role of the AE, responding to `ask` requests from the interface agent and applying the relevant rules from the rulebase to the domain information. User information is obtained either directly from the user model agent or passed down from the interface agent in the initial query. If the adaptation rules trigger an update of the user model then the adaptation agent will need to inform the user model agent of any new information.

7.4.3 Interface Agent

The interface agent, like the interface component, acts as the gateway between the user and the adaptation and user model agents. It converts requests by the user when they select a new page or ask for more information, into queries to the adaptation agent. The interface agent can also obtain user information and pass this onto the user model agent.

The flexibility of the agent architecture relies on the ability to deploy the interface agent in a range of locations. It is also worth noting that depending on the size and properties of a particular application, the interface agent might be broken down into a set of smaller agents and distributed across multiple platforms. In other situations, it might be advisable for there to be one interface agent for every user. In this instance, a personal interface agent would trail the user and provide a more personalised interface to the AHS.

7.5 Advantages of an Agent-Based Adaptive Hypermedia Framework

Having presented ABAH and discussed the role of each agent in the framework, there follows a discussion of the advantages this framework provides.

7.5.1 Flexibility

By setting the components of ABAH within an agent environment, the benefits of an agent-based architecture can be incorporated into this framework. Agents provide a versatile approach when designing real systems and they facilitate a variety of AHS implementations. Three components provide enough flexibility to ensure that the framework supports client-side, server-side and hybrid system designs and if it is demonstrated that all existing adaptive hypermedia systems can be categorised into one of these three architectures, consequently it can be deduced that the ABAH architecture can be used to implement any existing AHS.

Server-side

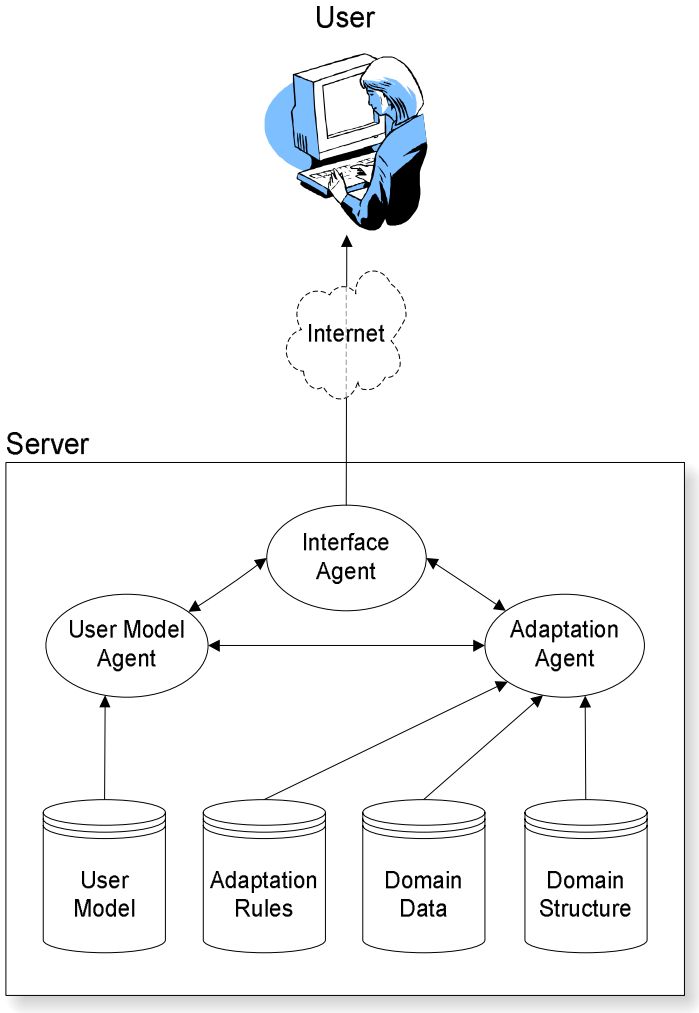

Figure 7-6. An example of a server-side implementation of ABAH

As shown in Figure 7-6, for the framework to accomplish server-side adaptation, each agent runs on a single web server and provides client access through standard HyperText Transfer Protocols (HTTP). This architecture was used by many of the early adaptive hypermedia applications. The user experiences little difference between an adaptive hypermedia site and a non-adaptive web site. All data, including all use models, resides on the server. This server-side arrangement of agents can be used to implement AHS like INTERBOOK (Section 2.6.1) and AHA! (Section 2.6.2).

Client-side

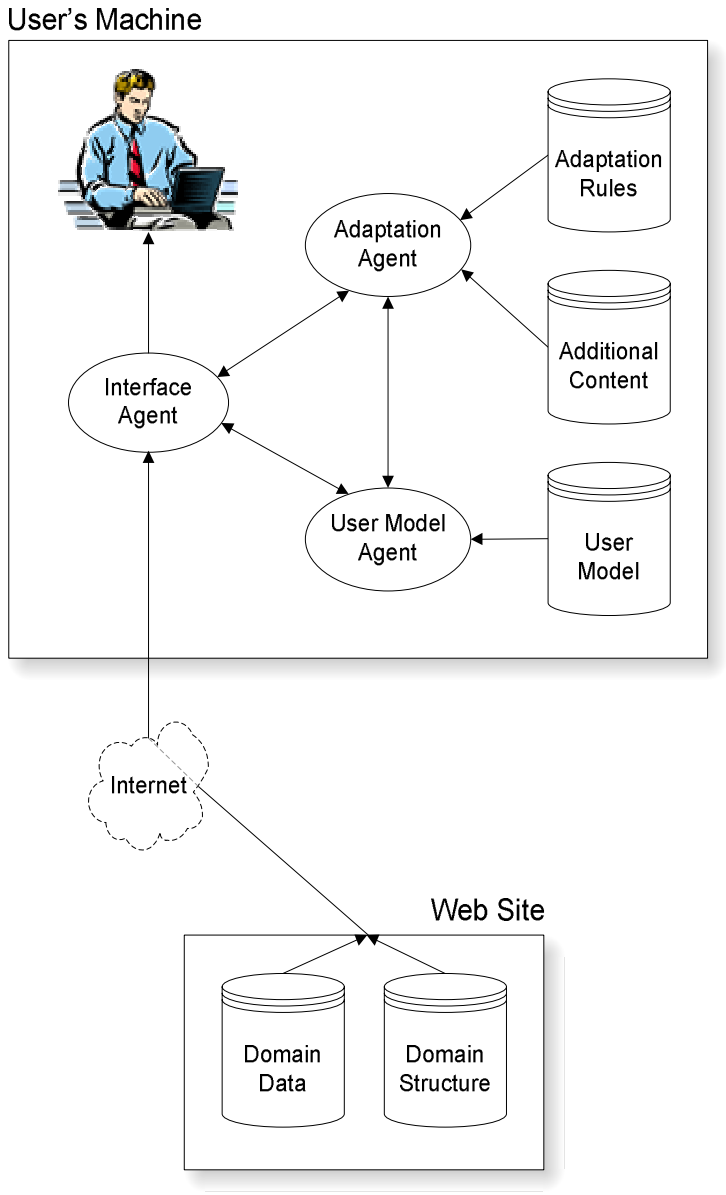

Figure 7-7. An example of a client-side implementation of ABAH

To implement the framework as a client-side application, the framework must run on the user's machine as shown in Figure 7-7. In this scenario all modelling data is stored locally and the interface agent has to intercept the information stream from the Internet, adapt it, and then display it to the user. The adaptation agent in this situation would not have access to the kind of resources available on the server, but instead it can analyse the web pages viewed by the user and provide adaptation using this profile and additional information provided either locally or from a separate source i.e. a third

party. PAADS provides a good example of a client-side AHS that uses this arrangement of components.

Hybrid Approach

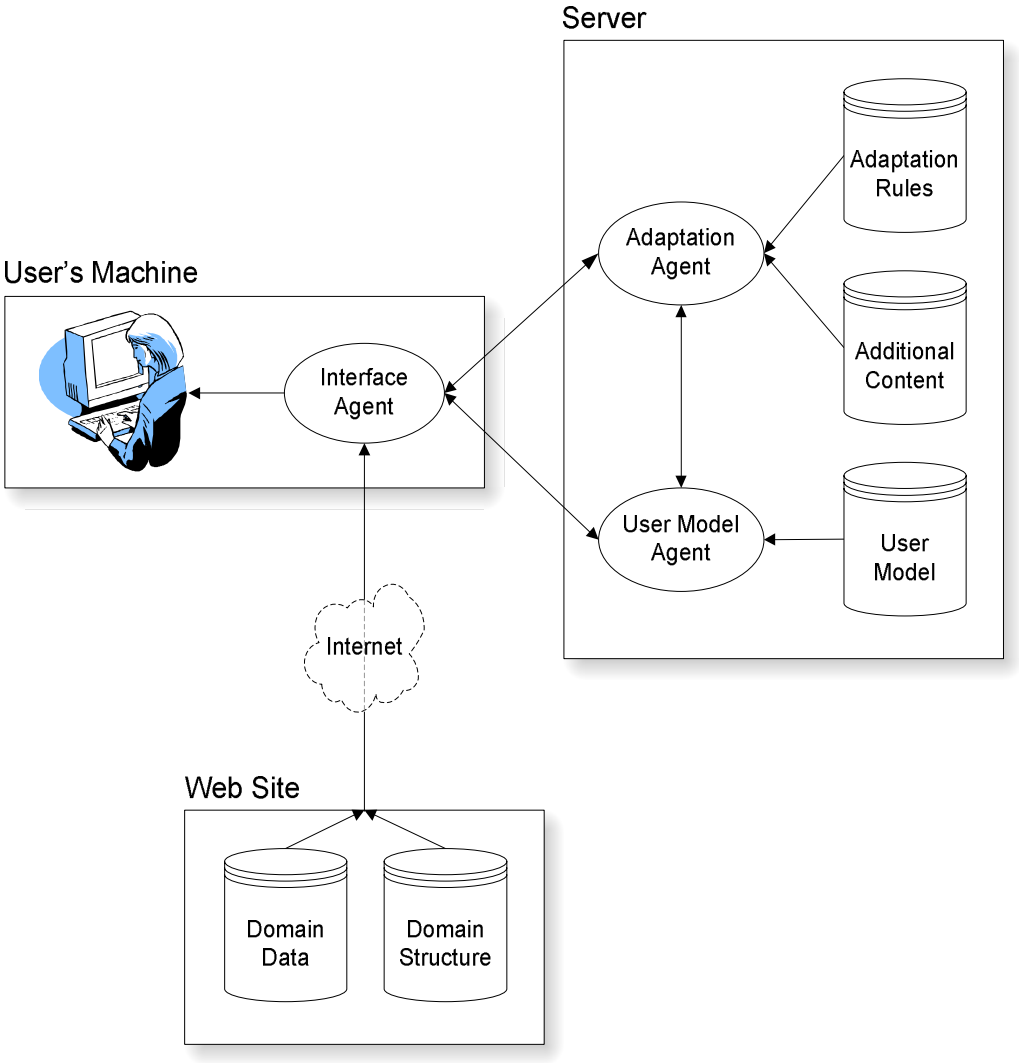

Figure 7-8. An example of a hybrid implementation of ABAH

It is also possible to arrange the framework as seen in Figure 7-8, to implement a hybrid system similar to the approaches used by systems such as Personal WebWatcher (Section 5.3.3) or WBI services (Maglio & Farrell, 2000). In this environment there is an interface agent running locally on the client machine while the remaining agents run on a separate server where they can be provided with a range of resources. This implementation provides the user with a personal interface agent, however their data is stored on a remote server.

An alternative approach would be to place the user model agent alongside the interface agent on the client's machine while the adaptation mechanisms reside on a separate server. This allows all profile information to reside locally while adaptation mechanisms can be maintained separately. If issues of security and access to personal information are important, this arrangement can help ease user concerns.

These three framework diagrams demonstrate the most logical arrangements for application design using this framework. The independent and potentially mobile nature of these agents allow many other arrangements to suite a variety of needs including possibility of having each agent run on a separate platform or even allow agents to migrate to different platforms depending on the set of available resources.

7.5.2 A New Definition for Adaptive Web-based Systems

Having presented the ABAH framework it is now possible to define the term adaptive web-based system from an agent perspective. While existing definitions concentrate on adaptive hypertext or hypermedia systems, this uses restrictive terminology that does not cover the plethora of adaptive web-based systems that are starting to appear. Many of these web-based systems are designed for new types of hardware coming onto the market that have restricted displays or limited processing capabilities. Examples include mobile phones (Cotter & Smyth, 2000) and Personal Digital Assistants (PDAs) (Oppermann & Specht, 1999)(Billsus *et al.*, 2000), areas that are expected to see much expansion in coming years (Alatalo & Peräaho, 2001).

The definition ABAH provides for an adaptive web-based system is as follows:

> *An adaptive web-based system is any web-based system that stores a model of the user's preferences and combines this with a set of adaptation rules to provide tailored services to clients.*

In this definition, the term *client* can be defined as any external entity either human or digital. As this definition is set within an agent environment, instead of providing adaptive information in the form of classical hypermedia pages, agent-based adaptive web-based systems provide adaptive *services* to clients. In a service-oriented

environment, there are a range of possible services that an adaptive web-based system could offer. Examples include:

- Discovering users with similar interests
- Finding products of interest to a user
- Creating personalised resources such as a personal newspapers
- Organise a user's schedule to best fit their personal requirements
- Locating local activities and events that match a user's interests

The definition of adaptive web-based systems presented here, while covering the existing range of adaptive hypermedia systems, also includes many of the activities assumed to be in the domain of intelligent agents. If a personal web agent is assigned the task of bidding for items, searching for the best deals or booking accommodation, it will need to keep a model of the user's preferences in addition to a set of rules for specifying actions under given circumstances. For example, if the agent is booking a flight ticket, and the user has stated that they require a business class seat, the agent must know what action to take if the specified flight has no available business class seats. Does it request an economy seat or a first class one? Does it try and book a different flight, or return and ask the user? The user's preferences and rules for governing which actions to take given the environmental conditions are all necessary components of intelligent agents, and as such these personal web agents should fall under the definition of adaptive web-based systems given above.

7.5.3 Open Architecture

Another feature of this framework follows the open hypermedia principle of offering link services to all applications. In ABAH, agents are free to respond to queries from outside the ABAH framework although this requires an agent framework that provides interoperability between agent systems (such as one of the FIPA-compliant frameworks). This allows both data and services to be used by adaptive and non-adaptive applications. Offering services in this way allows multiple applications to offer adaptive solutions to the same content, or use the information in different ways in a range of applications. This approach can increase competition between agents which can leading to bidding wars as the interface agent decides which agents to choose based on the features they each offer. The services provided by agents can also be made

accessible to other applications via some form of globally agreed web service model (discussed in Section 9.3). External applications would then be allowed to request user model information from the user model agent (providing they supply the required authentication), ask the adaptation agent for domain specific information, or even send messages to the user via the interface agent.

All of these features improve the openness of the framework, allowing both data and services to be accessed via external agents while formulating an environment that encourages competition and service negotiation between agents.

7.6 Summary

This chapter has presented the development of ABAH; the first agent-based framework for building adaptive hypermedia and adaptive web-based systems. The approach uses a set of three agents, each assigned the processing of a particular aspect of the system. It has been shown that by arranging these agents in various positions, the framework can support all current forms of AHS design; namely client, server and hybrid architectures. This chapter has also shown that using an agent approach to building AHS provides a range of additional benefits, from the flexibility of implementation, to helping produce an agent-based definition of adaptive web-based systems, and situating all this within an open environment that supports the outsourcing of data and services while promoting competition between agents. Following on from the development the ABAH, the next chapter evaluates this approach, using both implementation-based and theoretical studies to demonstrate and criticise some of the features of ABAH, and shows how it can be used in relation to formal models and existing systems.

Chapter 8

Evaluating ABAH

8.1 Introduction

There are two aspects to this evaluation phase, a practical and a theoretical aspect. The practical phase involves the design and construction of a server-side Adaptive Hypermedia System (AHS) based on the ABAH framework. This system incorporates the open hypermedia contextual link server, Auld Linky, as part of the adaptation mechanism to provide AH across an existing web site domain. The use of Fundamental Open Hypermedia Model (FOHM) objects in this system provides many interesting reflections on the implementation of adaptive techniques. The theoretical phase of this evaluation chapter looks at the relationship between ABAH and AHAM and describes how ABAH can be used as the implementation mechanism for AHAM. The second part of the theoretical phase consists of an analysis of a classical adaptive hypermedia system and describes exactly how this system could be written using ABAH agents.

8.2 Adapting Auld Linky: HA^{3}L

In the agent-based systems PAADS and ALIC, presented in Chapter 6, the open hypermedia concept of linkbases had been incorporated into their adaptation mechanisms. The idea of maintaining a separate domain structure model improves link consistency, encourages reusability and supports openness. The new system chosen to evaluate the ABAH framework incorporates the latest evolution of linkbase technology, Auld Linky (described in Section 3.3), into its adaptation agent. Together ABAH and Auld Linky are used to produce a medical web application that provides a wealth of adaptive features to support the user. This new system is referred to as Hypermedia Adaptation using Agents and Auld Linky (HA^{3}L).

HA^{3}L has several research aims.

1. To provide a server-side adaptive hypermedia system using ABAH agents.
2. To demonstrate the first use of Auld Linky to support adaptive hypermedia.
3. To reflect on the use of FOHM structures to support adaptive hypermedia techniques.

8.2.1 Overview of the Application

To evaluate ABAH, a new design approach was chosen to further demonstrate the flexibility of architectural approaches possible with ABAH. Whereas PAADS is primarily a client-side system and ALIC uses a hybrid architecture, this third agent-based AHS, uses a server-side approach where adaptation is provided across a single application domain. An existing medical domain was chosen to provide the content for this project. The data and organisational structure of this domain are encapsulated as FOHM structures and loaded into Auld Linky. Using the features of Auld Linky alongside the interface, modeller and adaptation agents present in ABAH, a range of adaptive techniques can be implemented to aid the user as they interact with the system.

The "JointZone" Rheumatology web site (Ng *et al.*, 2002) was selected as the application domain. It aims to encourage directed learning via structured trails through the information. The user profile will be generated by asking users to specify the type of trail they require, and their preferences about the trail. The system will then automatically generate the appropriate web pages and direct the user along the trail. Because the domain information only exists as FOHM structures maintained by Auld Linky (which requires an agent to dynamically generate each page), trails will therefore provide the only means of access to the material in the site.

8.2.2 Choosing a Suitable Domain for HA^{3}L

There are several requirements that need to be satisfied in choosing a domain for HA^{3}L.

- *Size* - The domain needs to be large enough to support the requirement of adaptation. If a domain is too small, then offering adaptation to the user will become unnecessary. This is because the information overload and lost-in-hyperspace problems need sizable domains for their effects to be experienced.

- *Structured Data* - The data from the chosen domain needs to be structured in such a way that allows it to be naturally converted into FOHM structures. As FOHM is designed to model a range of hypertext systems, most well organised hypertext data can be modelled with FOHM structures, however plain unstructured text, as found in the majority of web pages would require substantial manual effort to author the appropriate FOHM objects.

- *Relationships* - Any well-organised domain should have clear relationships and links between the various concepts within the domain to aid user orientation and increase their understanding of the material. Examples of these relationships include: subject hierarchies, topic examples, is-a relations and related information. In an open hypermedia system, information of this nature would be maintained separately from the data itself. It is the interconnected aspects of a domain that are easily specified as high-level FOHM structures such as tours, LoDs and concepts.

Given these considerations, a large well-structured domain called JointZone was chosen to illustrate the use of HA^{3}L. JointZone is a medical web site aimed at teaching degree-level medical students Rheumatology. Rheumatology is the study and treatment of rheumatic disorders (disorders that affecting movement of the musculoskeletal system). JointZone contains around 80 pages of text, images and video clips marked up in XML. Pages are grouped together into the various subjects within the field of Rheumatology. Each page contains information relating to a single topic. A page consists of an optional summary, introductory text and then a set of sub-topics. For example, in the ‘musculoskeletal examination’ section of the site, there is a page dedicated to examining the knee. This page contains a description of how to conduct the inspection, a small video showing a real examination, and then a listing of each possible knee-related condition including associated imagery. In addition to these pages, JointZone contains a large glossary of medical terms with over 380 entries. Each

entry contains a short paragraph of text describing the term and then a photo or picture if available. The only aspect of the system not incorporated into HA^{3}L are the 30 interactive case studies JointZone provides. Although these case studies could be used to further enhance the user model, they are unnecessary to fulfil the original goals of HA^{3}L and have therefore been left out.

JointZone uses the Extensible Stylesheet Language (XSL) and a set of Java servlets to render the web site and maintain the user models. JointZone evaluates the user's effective reading speed using a set of questionnaires and using this information, provides history-based link annotation for exploratory learning (Ng *et al.,* 2002).

Rather than manually process each of these pages, a set of Perl scripts were written to process the existing JointZone XML pages and convert them into the appropriate FOHM structures as shown in Figure 8-1.

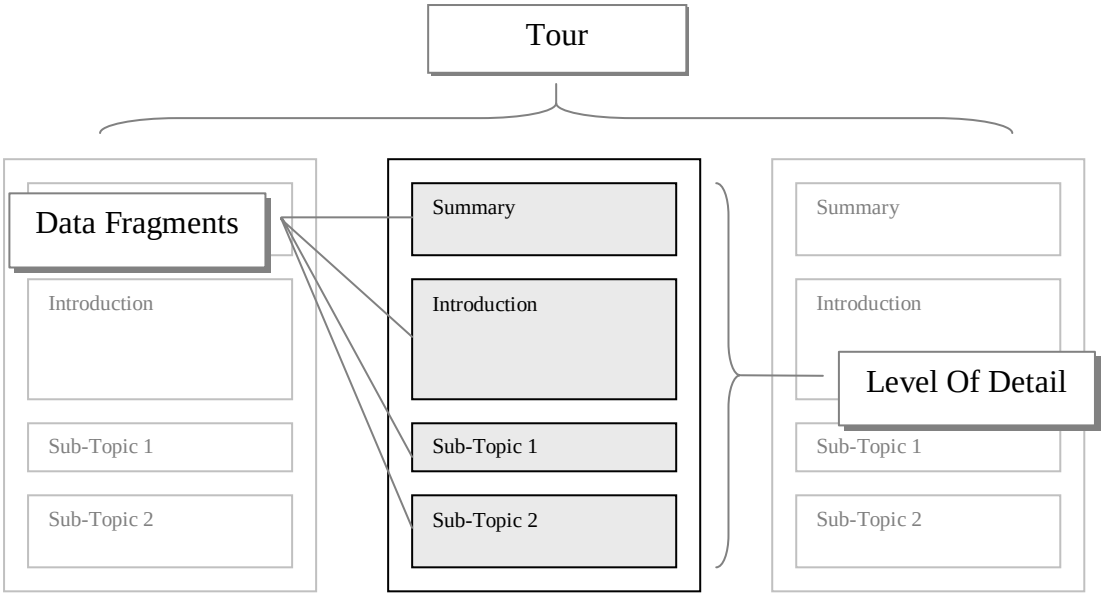

Figure 8-1. A diagram demonstrating JointZone pages expressed as FOHM Structures

Navigational Links are used to encapsulate glossary entries.

Data fragments are created for each of the sections that make up the page. Context is attached to the fragments in the form of a 'depth' field to indicate the complexity of the data item. Summary fragments are labelled 'basic', full introductions are labelled 'extended', while all additional information is labelled 'detailed'.

Behaviour objects are also associated with data fragments. There are behaviour objects representing the title of the fragment, the glossary items that appear within the content of the data item and a list of concepts obtained from the keywords in the fragment title. These behaviour objects are all associated with the event ‘display’ to indicate that the user model should be updated with the relevant properties when the user views the fragment. Figure 8-2 shows an example of a FOHM data fragment with behaviour objects attached.

```
<data id="investigations/blood_tests/biochemistry/urate">
  <behaviour>
    <event>display</event>
    <behaviourvalue key="title">Urate</behaviourvalue>
    <behaviourvalue key="glossary0">inflammation</behaviourvalue>
    <behaviourvalue key="glossary1">gout</behaviourvalue>
    <behaviourvalue key="concept0">urate</behaviourvalue>
  </behaviour>
  <context>
    <contextvalue key="depth">extended</contextvalue>
  </context>
  <datacontent><![CDATA[<P>An elevated plasma urate may be a pointer to gout as
a cause of joint inflammation but this result must be interpreted with caution
since hyperuricaemia is relatively common and may be totally
asymptomatic.</P>]]></datacontent>
</data>
```

Figure 8-2. An example XML FOHM Data Fragment

Level Of Detail (LoD) structures express the relationship between the summary, introductory text and any additional texts. There is one LoD structure for every page.

Concepts are used for situations where there is more than one media representation of the same data item. In the knee examination example described above, a concept would be used to represent both the video and text aspects of the introduction description. All concept associations in this application are constructed by hand.

A *Tour* is created for each subject area within the Rheumatology web site. The tour follows the existing structure used in JointZone. Tours string together the LoD objects to form trails through the Rheumatology web site.

The website therefore consists of a set of tours, where each tour links to a series of LoDs, and these LoDs point to specific data fragments.

8.2.3 The HA^{3}L System Architecture

The HA^{3}L system comprises all three ABAH agents and the Auld Linky server, arranged as shown in Figure 8-3.

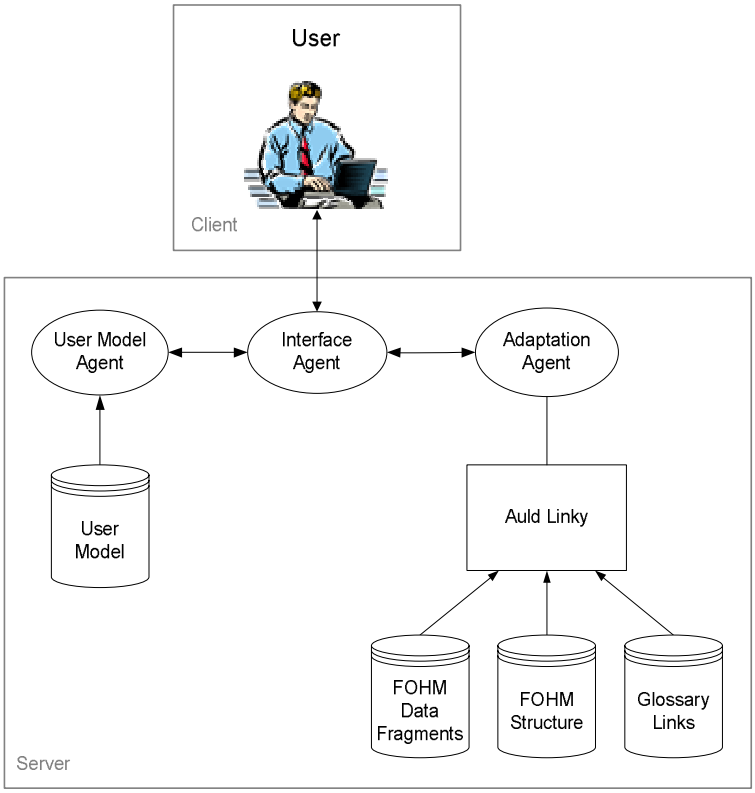

Figure 8-3. The HA^{3}L Architecture

The user model agent maintains a record of all the user's interactions with HA^{3}L. This includes the tours they select, the preferences chosen, the items on the tour they visit and any glossary links selected. Currently HA^{3}L makes no distinction between individual users. However, implementing an authentication scheme would be a trivial matter. The user model responds to queries from the interface agent requesting specific user information.

The adaptation agent acts as an interface between the Auld Linky server and the rest of the agent environment. It publicises a FOHM Ontology that other agents can use

to formulate FOHM queries. The FOHM Ontology was written to express FOHM objects within the SoFAR agent environment and conforms to a Document Type Definition (DTD), published by Auld Linky. The DTD ensures that the FOHM ontology provides compatibility with the contextual link server. When the adaptation agent receives a FOHM query, it converts it into the XML notation used by Auld Linky and sends it to the server. When Auld Linky responds, the adaptation agent converts the XML data back into the ontological representation and forwards the result onto the original requesting agent. In this way, the adaptation agent acts as a wrapper for Auld Linky, which takes on the role of the adaptation engine. The contextual link server maintains three domain models, marked up as FOHM structures, which together represent the entire Rheumatology domain. The data model contains all the data fragments, the structure model contains the associations between data fragments, and the glossary model is a linkbase of glossary items from glossary terms to the corresponding descriptions.

The third agent, the interface agent, facilitates communication between the agents and the user. The user interacts with the interface agent in the same way they would a web server. They explore the domain by selecting tours and clicking on links to new information, the interface agent converts these actions into requests to Auld Linky (via the adaptation agent) for the required information. The interface agent converts the FOHM information returned by the adaptation agent into HTML pages and sends this data back to the user's browser.

8.2.4 Adaptive Features

The system presents the user with a set of tours through the domain. There are tours for each of the different topics within the Rheumatology web site, e.g. 'Rheumatic Disorders', 'Disease Management' and 'Approach To Patient'. After the user has selected a tour, they can enter their preferences into the system. They can select between basic, extended or detailed versions of the tour, values corresponding to the context objects associated with each data fragment. The user can also choose to view the first paragraph of each data item and hide the rest via stretchtext. Stretchtext hides paragraphs, thereby visually shortening the length of each page allowing the user to quickly skim-read items on the tour. If they desire more information, the stretchtext items can be displayed at the click of a link. The final user preference is 'preferred

media type' for situations where there are concept objects describing multiple media representations. The choice of media type can be useful for instances where a user has limited bandwidth, a physical disability (such as blindness) or different styles of learning (listening to a speaker vs. reading a page of text).

When the user has chosen their options and requested a tour, the interface agent constructs the appropriate FOHM query and forwards it to the adaptation agent to query the FOHM server. The response obtained by the adaptation agent is returned to the interface agent, which organises the structures and constructs the web page. As the page is being built, the user model agent is queried for specific user information relating to the concepts and glossary items previously seen by the user; this allows the interface agent to provide link annotation and link hiding depending on the user's history. The following figure demonstrates how the tour will appear to the user.

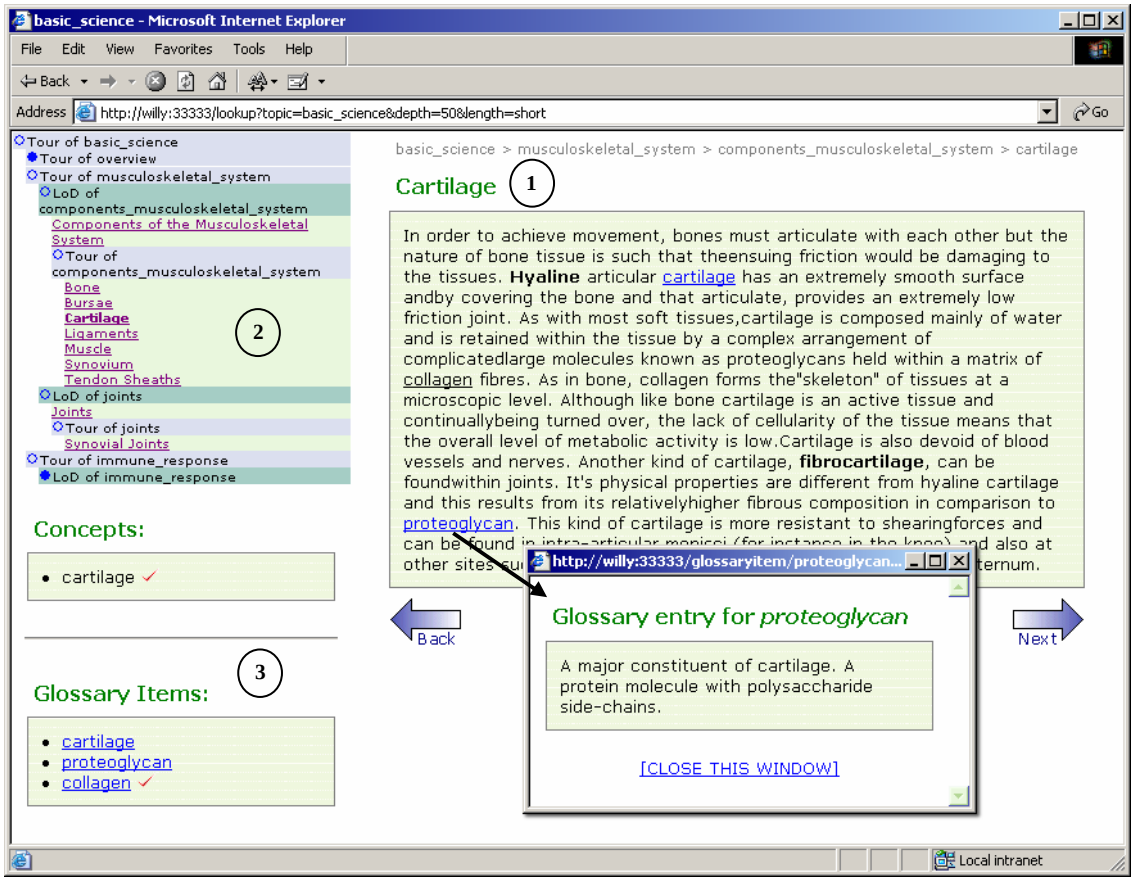

Figure 8-4. A HA^{3}L screenshot

As the user follows the tour or requests a glossary item, each user action is sent to the user model agent for storage. Using the profile and history data contained in the

user model and the context attached to Auld Linky's FOHM objects, HA^{3}L provides several adaptive features. These will be explained by a walkthrough of the user interface.

The interface is divided into three panels. The main interface panel (labelled (1) in Figure 8-4) displays the current data fragment on the tour. Each page may contain text, images, or video segments. At the request of the user, a stretchtext link will hide everything except the initial paragraph. Glossary items are rendered as hyperlinks within the text. If clicked on, the glossary entry for that keyword appears in a popup window (as indicated by the arrow on the screenshot in Figure 8-4). The interface agent records this action and passes the information onto the user model agent for storage. If the user visits a new page that contains the same glossary item, the glossary link is rendered in a different style. Cascading style sheets are used to provide a modified form of *link hiding* where the visited glossary items are rendered in the same colour as the main text but still underlined, however it would be a trivial matter to modify the interface agent to implement either *link disabling* or *link removal*. An example of this form of link hiding can be seen in Figure 8-4, where the glossary item 'collagen' is coloured black but still underlined.

At the bottom of the page are buttons for navigating forward and backward along the current tour. These buttons provide a form of *direct guidance* to the user. Additionally, if the user is viewing the first item of a LoD, then a 'More Information' button becomes active, allowing the user to iterate over each child node in the LoD. Iterating over the FOHM structures in this way mimics the custom depth control approach taken by MetaLinks (Murray, 2002). The system also provides a second form of *direct guidance* using automatic tour selection. At the request of the user the interface agent will query the user model and try and select a new tour previously unseen by the user. If the user has seen every tour, the system will then pick one at random. The user preferences for the tour, such as the depth of tour, stretchtext option and media preference, are chosen based on the frequency of previously chosen options. The automatic selection of media preference demonstrates a form of *adaptation of modality* in HA^{3}L.

The second numbered panel in Figure 8-4, labelled (2), provides a form of *map adaptation* to the user by displaying a hierarchical overview of the tour. The user's position in the tour (the current item shown in the main window) is highlighted on the map. The user can jump to any data item by clicking on the relevant link on the map. Each type of structure in the tour (LoD, tour or data item) is colour-coded on the map to aid identification.

The third information panel (3) in the above screenshot presents the user with a list of concepts and glossary items associated with the current item displayed in the main window. Both the concept and glossary information elements use *adaptive link annotation* by placing a tick mark next to the concept and glossary items the user has already seen.

8.2.5 Discussion

This section initially presented three research goals for HA^{3}L. After describing how the system was implemented, it is possible to re-evaluate those goals. It is worth noting that the implementation of HA^{3}L, given the programmer's experiences of PAADS and ALIC, took two weeks to complete. Such a short period of time is testimony to the ease of writing agents and their applicability as tools for rapid application development. There follows a discussion of how HA^{3}L met the original aims specified.

Aim 1: Provide a server-side adaptive hypermedia system using ABAH agents.

HA^{3}L provides an example of an adaptive hypermedia system that has been written using ABAH agents. This adaptive server-side architecture provides the third approach to adaptive hypermedia systems using agents, complementing the client-side PAADS and hybrid approach of ALIC. Together they demonstrate the flexibility of implementing AH applications with agents.

Aim 2: Demonstrate the first use of Auld Linky to support adaptive hypermedia.

Although the JointZone domain information was stored as FOHM objects and accessed using Auld Linky, this open hypermedia approach does not offer the only solution. The adaptation agent could store all domain data using unstructured text files, or insert every data item into a database and use complex database queries to retrieve

the information. However, these solutions do not offer the same level of functionality as provided by Auld Linky and would require additional code to implement the same set of features.

The decision to use Auld Linky also demonstrates the expressive nature of FOHM. Auld Linky has already been used within augmented reality environments (Sinclair, *et al.*, 2002) and as a tool for building narrative structures (Weal *et al.*, 2001). HA^{3}L provides the first example of the use of Auld Linky within an agent environment and specifically as a mechanism for constructing adaptive hypermedia web sites. While the expression of a domain in FOHM has implications for increased interoperability between AHS, it is the inclusion of context and behaviour objects and the culling process employed by Auld Linky that truly aids adaptation. Behaviour, attached to data and associated by means of a trigger event, is an ideal method for building up a user model and can be used by clients of Auld Linky to keep a record of user activities. Unlike Behaviour objects, Context objects are processed by Auld Linky and used to confine a hypermedia space, limiting the number of possible user actions and therefore providing a personalised view on a wider hypermedia environment. Within an adaptive environment, context culling can provide a list of possible user destinations, given the user's current contextual state. Using additional information from user models, clients of Auld Linky can select between these possible destinations. Together, the features of FOHM and Auld Linky have many applications within the field of adaptive hypermedia and it is hoped that HA^{3}L has initiated the process of developing these.

Aim 3: Reflect on the use of FOHM structures to support adaptive hypermedia techniques.

The adaptive features that HA^{3}L provides cover a large part of the adaptive taxonomy developed by Brusilovsky. The wide variety of adaptive techniques implemented in HA^{3}L were chosen as a means to investigate the possibilities of using FOHM objects to represent the data structures upon which adaptive features can be built. This led to an over use of adaptive techniques, resulting in a system that was never designed for user evaluations. Because of this, there can be no assumptions made about the quality or effectiveness of the adaptive features. Instead, HA^{3}L allows a reflection on the fundamental mechanisms for providing adaptive hypermedia. The use of FOHM's navigational links, data fragments, concepts, tours and LoDs in HA^{3}L has

shown how many of the adaptive techniques coined by Brusilovsky have similar underlying structural representations. This is best explained with the use of Figure 8-5 which demonstrates how the various FOHM objects can be used, individually or together, to implement each adaptive technique.

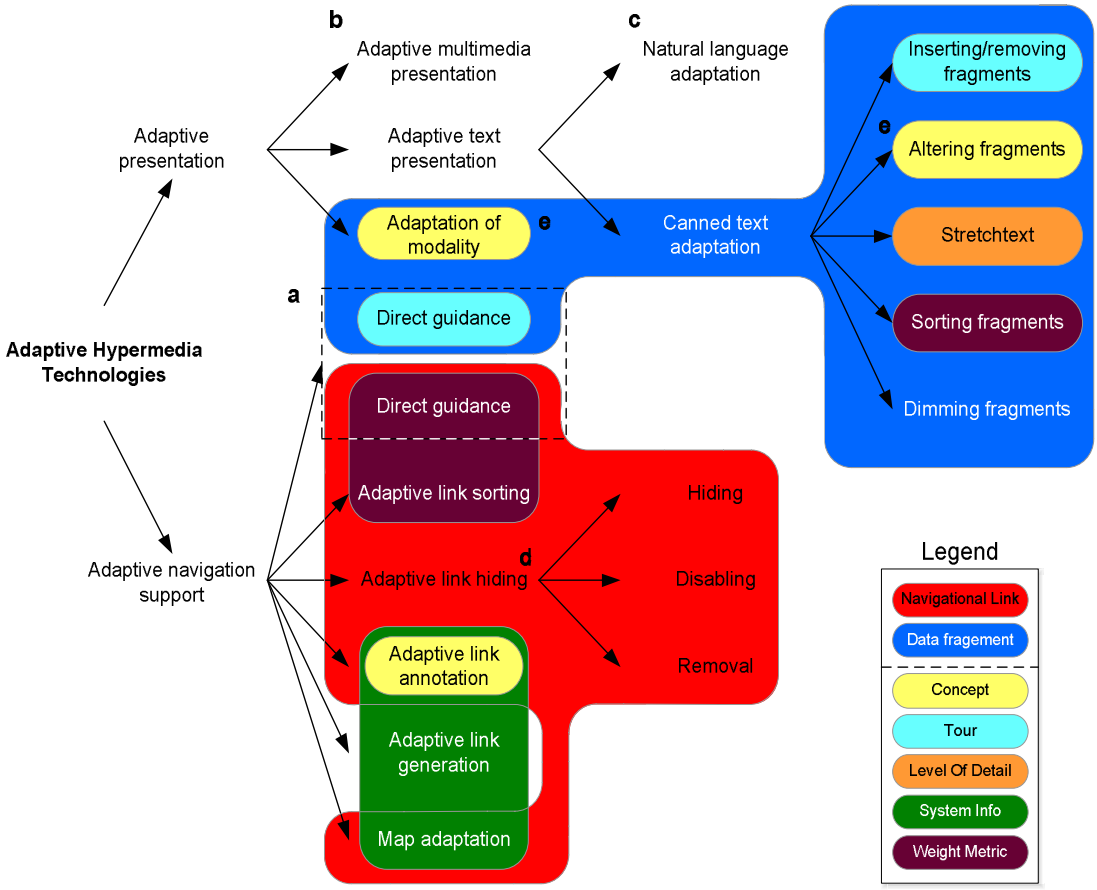

Figure 8-5. A structural perspective on Brusilovsky's taxonomy

Each technique in the taxonomy is built up using the structures or resources required to implement that technique. For example, 'Adaptation of modality' would require multiple data fragments, each representing the same information but using different media types. Concept structures would then collect these fragments together and the AHS would choose the media fragment to present to the user based on user preferences.

This diagram raises several important issues (labelled a-e in Figure 8-5) concerning the taxonomy that deserve further explanation.

(a) The only visual change to the taxonomy from Brusilovsky's 2001 original, involves duplicating 'Direct Guidance' as it appears this technique has two possible implementations. The first approach would use a tour of the information domain; the adaptive 'next' button would therefore point to the next appropriate destination on the tour. A second way of implementing direct guidance is to sort all available link destinations (requiring a weight on each link) and then select the first link, the one with the highest weight, as the next destination. Each approach has been given a separate representation on the diagram.

(b) One of the philosophies behind open hypermedia modelling is a deliberate disregard for specific data formats and media types. This disregard for data types provides a new perspective on Brusilovsky's taxonomy. It should be apparent that any of the 'Adaptive text presentation' techniques described in the taxonomy can be applied equally well to 'Adaptive multimedia presentation'. To improve clarity on the diagram, both techniques could be combined and re-labelled 'Adaptive Media Presentation'.

(c) This perspective also questions the way that Natural Language (NL) Adaptation is located as a sub-branch of text adaptation. Because NL adaptation should not be limited to purely text-only media (for example, there is no reason why audio clips could not be use in place of text), a better alternative would position NL adaptation under a larger umbrella of 'Adaptive Sequencing', which might use canned or constructed information fragments. Another issue with NL adaptation is how it influences nearly every single technique in the taxonomy. Whenever fragments of information are stitched together, there is a need to conserve the progression of the narrative flow. In these situations, sequencing methods like NL techniques can be used to merge fragments together. This raises the issue of whether the inherent nature of NL techniques requires a specific label on the diagram.

(d) The fourth issue raised by the diagram is the consideration that the various subcategories of 'Adaptive link hiding'; hiding, disabling and removal, are all structurally equivalent. This means that a system using navigational link objects to implement one adaptive link hiding method has all the features needed to implement any of the other adaptive link techniques. While link hiding and disabling can easily be accomplished by modifying style sheets, link removal requires specific client support.

However structurally speaking, each technique is using the same objects for their representation of links.

(e) The diagram also shows the apparent similarity between 'Adaptation of modality' and 'Altering fragments'. These two techniques are functionally identical if it is considered that fragments can contain multiple media representations of the same data objects. In such cases, choosing the best media type to display (adaptation of modality) is the same process as selecting one fragment from a set of fragments (altering fragments).

This thesis will not attempt to re-write Brusilovsky's taxonomy or invent a new one. However it is hoped that the use of FOHM structures, as shown in Figure 8-5 and implemented in HA^3L, will give application designers new insights into the type of structures needed to implement the various adaptive techniques. For instance, a system that uses adaptive link sorting should also be able to provide a form of direct guidance without adding any new data structures to their applications. This work also provides the foundation for building an interchange format for adaptive hypermedia systems. By specifying the fundamental structures present in each adaptive technique, it is possible to start designing a language that systems could use to export the structure of their domains. It is hoped that FOHM can provide the starting point for such future interoperability research.

8.3 Comparisons with AHAM

The second part of the ABAH evaluation involves a theoretical discussion on the role ABAH plays with AHAM, and details a possible approach to combining these two techniques. Because of the similarities AHAM shares with the Munich reference model, this section applies equally well to both models.

The AHAM reference model (described in Section 2.7.1) provides a mechanism for describing the functionality of AHS at the conceptual level (Wu *et al.*, 2001). In much the same way as the open hypermedia protocol provided the first interoperability standard for hypermedia systems, AHAM attempts the same thing for the field of adaptive hypermedia. As an abstract representation of an adaptive hypermedia system, it does not specify the implementation specifics nor define the format and structure of

the information contained in the system. AHAM was designed for several reasons. Firstly, it allows developers to specify their systems using the terms and concepts of AHAM, providing a common ground to analyse the similarities and differences between AHS. Secondly, it provides a formal language for developing new AHS making clear distinctions between the various data models used in adaptive hypermedia systems.

The agent framework presented in this document represents one method of implementing an AHS expressed in AHAM. Although both AHAM and the agent framework concentrate on different aspects of system development; requirements analysis in AHAM and system design in ABAH, the framework parallels many of the ideas developed in AHAM, thereby simplifying the process of moving from AHAM-expressed concept to fully agent-implemented system. All of the components within the agent framework can be easily mapped onto the various levels of the AHAM model, as shown in Figure 8-6. This demonstrates that it should be possible to implement every system defined in AHAM using ABAH.

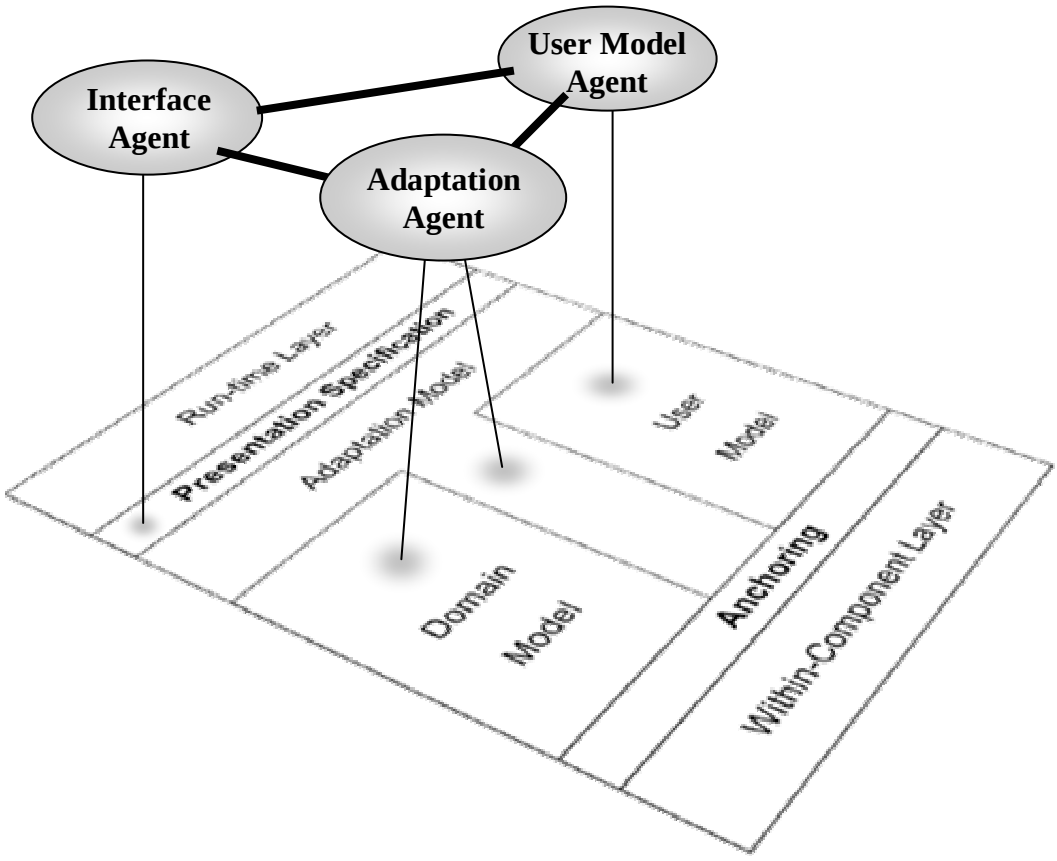

Figure 8-6. Roles of AHAM components within the Agent Framework

Once a system has been expressed in AHAM, a developer can design the agent framework to encapsulate each aspect of the model. Both domain and user models will be represented as databases of information managed by the agents. The adaptation agent will apply the rules formulated in the AHAM model to the data within the domain model and query the user model agent for specific profile information. The interface agent will apply the presentation specifications to the domain information after the rules have been processed, and present this information to the user.

The parallel between AHAM and the agent framework can be further enhanced by developing a specific AHAM instance of the agent framework. Currently the agent framework provides a guideline for the transition from concept to implementation, however it would be possible to design a set of agents specifically aimed at implementing AHAM models. Domain information would follow the component, anchor and within-component aspects of the Dexter model. The user model agent would understand the attribute-value pair structure of the AHAM user profile, and the adaptation agent applies rules specified in the rule syntax of AHAM to the domain information. The interface agent uses the presentation specifications stored alongside the domain information to render the page to the user. With dedicated agents such as these and authoring tools for the various AHAM models, a revolution might be brought about for adaptive hypermedia system development. Designers need only conceptualise a system in AHAM, use tools to author the domain, presentation and adaptation rules and then tailor the interface agent to the particular needs of that system. Although currently no authoring tools have been developed for AHAM, having these would provide all the necessary tools for a complete adaptive hypermedia Software Development Kit (SDK).

An SDK that relies on AHAM will be limited to developing adaptive hypermedia systems, rather than adaptive web-based systems. This is due in part to the reliance on the Dexter model for *hypertext* systems, and partly due to the limitations of the Dexter model itself, which is over a decade old. Aside from the inability to cope with spatial or taxonomic hypertext, one of the limitations of the Dexter model is the lack of support for temporal media (Hardman *et al.,* 1994), specifically concerning the timing and synchronisation of multimedia data. As adaptive hypertext systems become reliant on a wider variety of media, this will become an important issue for future adaptive

hypermedia systems. The reliance on a hypertext model is also a limiting factor with the emergence of the semantic web and web services which will bring a much more dynamic nature to the information being adapted (see Section 9.3). Discovery of new services and the inference of new information is a difficult modelling task as the nature of the information is unknown at design time. While AHAM is sufficient to model the existing generation of AHS, it will need major modifications if it is to keep up with the continual evolution of adaptive web-based systems which are moving away from traditional hypertext models.

When considering AHAM, it is also worth mentioning AHA! As an example of a specific instance of AHAM, AHA! provides a range of features for building adaptive web sites. As mentioned at the end of Section 3.4, there are several similarities between AHA! and the contextual link server Auld Linky. Both maintain (structured) information and depending on values in a user model they hide or show (cull) information before presenting it to the user. While on a functional level, both offer a very similar set of features, there are clear implementation differences between the two. While AHA! is a dedicated system for developing adaptive hypermedia applications, Auld Linky is a highly complex FOHM server. To emulate the features of AHA!, Auld Linky would require a specifically written client to store the user model data and perform the FOHM queries. Although the unnecessary complexity in Auld Linky might limit its desirability as a vehicle to emulate AHA!, it would make interesting further research to compare these two approaches; the adaptive hypermedia application development system, and the open hypermedia contextual FOHM server.

8.4 Modelling INTERBOOK with ABAH

ABAH can also be used to implement existing adaptive hypermedia systems. For example the classic INTERBOOK application has many similarities with the HA^{3}L system. Both are server-side systems and present a similar interface to the user with a main window, a glossary window and a spatial map of their location within the domain. They also both provide adaptive link annotation and different forms of direct guidance. Similar mechanisms for web-based interaction are employed by each system and they both use structured domain models for representing knowledge. However, their differences are equally apparent. HA^{3}L supports more adaptive techniques such as adaptation of modality and all forms of link hiding, while INTERBOOK stores concept

pre-requisite information which leads to a more advanced user model. INTERBOOK tracks three states for each concept; learned, ready to be learned or not ready, this is compared to HA^{3}L's binary, seen or not seen concepts. Both systems also adopt different learning styles. INTERBOOK's domain model is structured around a book metaphor and provides exploratory learning of any of the units within the book. HA^{3}L's trails are primarily a mechanism for directed learning with the user choosing which topic or trail to follow.

To achieve an ABAH implementation of INTERBOOK, a similar architecture to HA^{3}L could be used. The major addition would be including pre-requisite relationship information between various concepts. To realize this, changes are needed in the FOHM structures, however the expressive nature of FOHM makes these changes easily implemented. While the approach in HA^{3}L was to store a depth indicator in the context field of each data fragment, this could be replaced with multiple pre-requisite fields indicating which data fragments a user would need to have read before being able to view the next one. Using this approach, if Auld Linky was queried for destinations from their current position given their viewing history, the server would only return those destinations that the user would be capable of understanding. However, currently INTERBOOK displays all locations and annotates those unadvisable for the user. An alternative approach that would more closely mimic INTERBOOK would be to locate the pre-requisite information in the behaviour values of data items, as demonstrated in Figure 8-7. Instead of being culled by Auld Linky, the server would return all possible destinations and then the interface agent could annotate those fragments where the behaviour value does not match the user's concept model.

```
<data id="investigations/blood_tests/biochemistry/urate">
  <behaviour>
    <event>display</event>
    <behaviourvalue key="title">Urate</behaviourvalue>
    <behaviourvalue key="glossary0">inflammation</behaviourvalue>
    <behaviourvalue key="glossary1">gout</behaviourvalue>
    <behaviourvalue key="concept0">urate</behaviourvalue>
    <behaviourvalue key="prerequisite0">plasma</behaviourvalue>
    <behaviourvalue key="prerequisite1">hyperuricaemia</behaviourvalue>
  </behaviour>
  <context>
    <contextvalue key="depth">extended</contextvalue>
  </context>
  <datacontent><![CDATA[<P>An elevated plasma urate may be a pointer to gout as
a cause of joint inflammation but this result must be interpreted with caution
since hyperuricaemia is relatively common and may be totally
asymptomatic.</P>]]></datacontent>
</data>
```

Figure 8-7. Example of FOHM behaviour for storing pre-requisite information

The addition of pre-requisite information is the largest change needed to make HA^{3}L an agent-based alternative to INTERBOOK. The second change requires converting the limited user navigation options supported by HA^{3}L into an exploratory learning environment. This could be done by querying Auld Linky for all possible first-level destinations and then present them to the user. INTERBOOK-style adaptive link annotation can be given by comparing the behaviour pre-requisite information on each data structure to the contents of the user's model. Visited structures can be coloured white, data structures with pre-requisite concepts already visited by the user can be coloured green while all other links can be coloured red. This would mimic INTERBOOK's traffic light metaphor (Figure 2-3). Most other changes are cosmetic in nature and are relatively simple to implement.

8.5 Discussion

INTERBOOK is just another example of the range of existing adaptive hypermedia systems that can be implemented using ABAH. INTERBOOK is closely related to the ELM-ART range of systems which add online exams and testing to the students learning experience. If the ABAH adaptation agent were equipped with a

mechanism for producing and marking tests, there is little reason why ABAH could not be used to implement systems of a similar nature to ELM-ART.

While ABAH might offer solutions to developers who need to implement a system designed using a theoretical model, or need to choose an implementation that provides a high level of flexibility, it should not be offered as a global solution to solve all problems. There are instances where ABAH would provide an over-engineered solution to a problem. One of conditions that ABAH enforces on an application is a high level of communication between components. While this might be perfectly acceptable for many applications, it could impose unnecessary overheads on smaller systems where a better solution would be to combine the agents and write a single program. This would simplify many aspects of the system but there is a danger that programs become monolithic, obfuscated and difficult to upgrade.

The decision to use an agent architecture introduces its own set of disadvantages for system designers. These disadvantages are not limited to ABAH and apply to any system built with such a technology. Aside from the unpredictability of agents (Wagner, 2000) mentioned in Section 4.3, another problem with agent systems is the large overhead imposed by the agent environment within which the agents execute. These environments consume resources and impose large installation and setup costs. An analogy would be with the Java virtual machine. Although large-scale applications can be written in Java, the trade-off for cross-platform support has meant that a virtual machine imposes overheads as it interprets each line of [byte] code, and this causes the slower execution speed when compared to non-interpreted languages such as C or C++ (Niemeyer & Peck, 1997). Although not all agents need to execute within an agent environment, a noticeable exception being JatLite as discussed in Section 4.6.1, most systems employ them to take full advantage of agent properties. Using agents might be acceptable on larger server-side implementations, however it is most likely to cause inconvenience with client-side adaptive hypermedia and web-based systems where small, simple, low cost applications are desirable. For example, the approach in PAADS where agents run on users' machines is unlikely to win favour with real users due to the large installation, setup and running costs of the SoFAR framework. However it is worth noting that the development of hardware (and in turn software) continues to follow Moore's Law (Moore, 1965) and as such, it is likely that overheads

such as these will have less impact on the decision to use agent frameworks in the future.

Another impedance to choosing ABAH is the current state of agent research. Typically the field of agents has long been the place of debate about the definitions, applications and approaches to developing agents (Wooldridge & Jennings, 1995)(Nwana, 1996). Deciding to use agents in a system is largely a matter of faith, as assumptions have to be made that the chosen agent framework will persist over time and that developers can continue to receive technical support. While the field of agent research has many tens of agent frameworks in development, it has equal numbers of abandoned and outdated frameworks. It is hoped that standards like KQML and FIPA can help alleviate some of these concerns with new standards for communication and interoperation. Although the future is undecided, it is clear that agent research has yet to come to fruition.

8.6 Summary

The choice to use ABAH as the foundation for building agent-based AHS will ultimately be a personal one for the developer. HA^{3}L provides an example of how ABAH can be used to rapidly produce adaptive hypermedia applications. Like PAADS and ALIC before it, ABAH demonstrates unique agent approaches to creating a variety of adaptive applications. The theoretical sections of this chapter have examined techniques for using ABAH to realise AHAM models, and discussed some of the limitations of the AHAM approach in general. There has also been a look at how ABAH can be used to implement existing AHS, such as INTERBOOK. The penultimate section of this chapter has talked about the disadvantages of using ABAH, including situations where a framework such as ABAH would be unnecessary, and the difficulties of committing to agent technology.

The following final chapter concludes with a summary of the work in this thesis and presents the possible future directions for extending the ABAH framework.

Chapter 9

Conclusions

This thesis has presented ABAH, an agent-based framework for implementing adaptive hypermedia applications. This framework accommodates the major components of existing Adaptive Hypermedia Systems (AHS) by constructing agent roles for each of them. Setting this framework within an agent-based environment is not only unique in adaptive hypermedia development, but it brings a range of features from the agent world that can aid development and enhance the functionality of AHS.

9.1 Summary and Conclusions

The majority of existing adaptive hypermedia systems have been built with traditional application-oriented techniques, rarely achieving large-scale domain independence. Recent years have seen a maturing of the field through developments such as Brusilovsky's taxonomy which attempts to categorise AH techniques, and modelling techniques presented in Section 2.7. However there has been little opposition or alternative approaches developed to challenge these areas of research. It is not the role of the framework presented in this work to confront or compete with these earlier approaches, instead ABAH presents a new approach for the adaptive hypermedia community. The framework proposes that agent-based systems are well suited as the foundation for building adaptive hypermedia and web-based systems. This has been demonstrated in the successful development of three agent-based systems; PAADS, ALIC and HA^{3}L. Table 9-1 summarises these three systems.

SYSTEM	ARCHITECTURE	ADAPTIVE TECHNIQUES SUPPORTED	BASIS FOR USER MODEL	NOVEL FEATURE
PAADS	Client-Side	Augmented Linking	Keywords extracted from the titles of web pages	Agent-Based AH
ALIC	Hybrid	Contextual Augmented Linking	Clusters of TFIDF vectors from previously visited web pages	Context-dependent links
HA^3L	Server-Side	Direct Guidance Adaptive tour selection Link hiding Link Annotation Map Adaptation Stretchtext Adaptation of Modality	Concepts from visited data items Tours the user chooses Options chosen for these tours	Using Auld Linky to provide Adaptive Hypermedia

Table 9-1. A comparison of the adaptive hypermedia systems: PAADS, ALIC and HA^3L

In addition to the role agents can play in implementing AHS, the framework also provides a host of additional features to system designers and developers. These benefits include:

Reduction in application development time - Most agent architectures provide developers with tools for decreasing production time. For instance, the SoFAR agent environment supports developers by providing a syntax for defining the properties and attributes of agents which can be compiled into base classes and then extended though Java inheritance functions. Ontologies also have a similar description syntax for compiling the appropriate classes. SoFAR, like JADE, JATLite and many other agent frameworks, abstract the communication processes away from the developer allowing them to concentrate on functionality without being concerned with implementation-specific issues such as message formats, agent discovery and method invocation.

Flexibility - Chapter 6 has shown some of the possible arrangements that the agent framework can be organised into. The framework can be completely distributed, centralised on one server or allowed to reorganise itself into the best possible configuration providing its agents have mobility.

Reliability - Agents operate in an environment that can be continually changing and as a result they need to be able to cope with a wide variety of conditions. This produces reliable agents which can react to changes in the environment and survive conditions that might otherwise lead to catastrophic system failures. For example, all

three agent-based systems developed for this research will continue to operate (albeit in a limited way) if any of the agents fail for a particular reason.

Service-oriented approach - Agents are service-oriented components, executing in environments where new services can easily be advertised, discovered and invoked. Services bring new approaches to programming. Competition is encouraged between agents, leading to possible scenarios where adaptation agents bid for contracts to provide users with specialised features. A user interface agent could discover new services and selectively chose which ones to offer to the user. With appropriate security in place, the user model agent could cooperate with other services to provide user information while keeping a single storage location. Users would not need to keep re-entering their preferences, instead programs would automatically discover the location of the user model agent and make use of profile details such as available bandwidth, location, gender, etc.

This work has also benefited from a strong foundation in hypermedia and a continual influx of open hypermedia concepts and ideas. One of the early influential systems, Microcosm, the forerunner to much of this research, contained many modern concepts, like the separation of links from documents and agent-like chains of filters to provide adaptive features. The more refined distributed linkbase approach apparent in the DLS and which heavily influenced the first open hypermedia protocol has been integrated into each system constructed for this research.

The interplay with open hypermedia concepts is most apparent in HA^3L, which incorporates the Auld Linky contextual FOHM server into the agent framework to provide adaptive hypermedia. The system demonstrates how FOHM's data structures can be used to implement many of the adaptive hypermedia techniques. However although Auld Linky provides many powerful mechanisms, it lacks a weighting metric which is needed it if is to implement the adaptive sorting techniques. If the binary matching routine were to be replaced with a weighted one, Auld Linky could then indicate the degree to which a query matched a structure. The server could then provide the first attempt towards an interchange format for AH, with FOHM objects used to standardise the representation of domain information.

There are clearly areas for crossover between the fields of open hypermedia and adaptive hypermedia. Both groups originate from the Hypertext community and while each tackle a different aspect of hypermedia systems, this thesis demonstrates some of the areas where AH can incorporate concepts developed by the open hypermedia community.

9.1.1 Novel Research in this Thesis

This thesis has presented an agent-based framework for developing adaptive hypermedia applications. The contributions this work makes in the field of adaptive hypermedia is summarised in the following list:

- The development of a new framework for adaptive hypermedia and adaptive web-based systems.
- A justification for the decision to base this framework on agent technology.
- A new definition for general adaptive web-based systems that includes, but does not limit itself to, adaptive hypertext systems.
- An examination of the crossover between the fields of adaptive hypermedia and open hypermedia, including a critical analysis of the popular taxonomy of adaptive techniques.
- The demonstration of the first use of Auld Linky, a contextual link server, to provide adaptive hypermedia.

9.2 Future Work

While the framework provides many new features and benefits to AH developers, this section discusses some of the possibilities for extending this work.

Implement an existing adaptive hypermedia system - Although ABAH is supported by the practical and theoretical evaluations in the previous chapter, there is a clear need to use ABAH to replicate an existing adaptive hypermedia system. Following the theoretical discussions concerning AHAM and INTERBOOK in Sections 8.3 and 8.4, implementing a system such as INTERBOOK would help prove the worthiness of this agent-based approach.

Further evaluations - While ABAH offers many features to system designers, there has been no work to indicate exactly to what extent this would aid developers. While ABAH may be used to produce a system that looks and feels exactly the same as other AHS, this isn't always a practical or desirable approach. After ABAH has been used to implement an existing system, such as INTERBOOK, it would be possible to evaluate these two approaches and provide a basis for an in-depth comparison of techniques.

Additional tools - As mentioned in the discussion of ABAH and AHAM (Section 8.3), there is clearly a need to develop a set of agents that can recognise AHAM structures and process AHAM rules. Providing an AHAM-aware agent-based framework would facilitate rapid application development and reduce the amount of coding necessary. System designers can formally specify their systems in AHAM using the variety of existing modelling techniques. When stated in this way it would be a trivial matter to use a yet-to-be written AHAM authoring package to create the domain material, links and rules which could immediately be understood by the agent framework. To develop the complete adaptive hypermedia system using this approach, programmers need only write a specific interface agent to suit their requirements. Agent-supported AHAM tools such as these would strengthen the design and development process, and provide the first steps towards a software development toolkit for AH.

Merger of OH and AH - Another further topic for research is to explore the interrelation between the open and adaptive hypermedia research fields. Recent work suggests that the adaptive community is just starting to examine these issues (Henze & Nejdl, 2002)(Bailey *et al.*, 2002) by using the concepts and philosophies developed in the open hypermedia community to reflect on adaptive hypermedia work. There are already several adaptive applications being considered for Auld Linky. The growth of the adaptive hypermedia community in recent years is beginning to impact on the other hypertext-based communities. As people realise that the future for traditional hypertext systems is to empower them with user models and adaptation, it is hoped that adaptive hypermedia research will start looking at bigger issues such as interoperability, communication between systems, interaction mechanisms and the role that adaptive web-based systems, will take in the future.

9.3 The Future of Agent-Based Adaptive Hypermedia Systems

The last decade has seen the explosion of the World Wide Web, the creation of both the adaptive hypermedia and open hypermedia communities, and the increase in popularity of agent technology. The majority of the work in these areas has been produced by academic research departments. However, the visions of key influential people and large businesses are slowly transforming the landscape, promising to totally revolutionise the existing web. These changes will have a large impact on both the agent and adaptive hypermedia communities.

Agent Technology

Agents play a multitude of roles in today's computer systems, ranging from online bartering, personal assistants and satellite navigation, to new paradigms for programming, ubiquitous computing, knowledge management and vessels for intelligent systems. Agents have a wide range of applications, and while the provision of adaptive hypermedia is just one aspect, systems employing agents will be able to take advantage of the features and services provided by other agents. As the field continues to develop, agents will take on a much wider role, possibly becoming the new software engineering unit (Jennings, 2000)(Petrie, 2000). Although agents have yet to take over the world as the hype of the early and mid nineties would have suggested (Janca, 1995)(Guilfoyle & Warner, 1994), the importance of agents for the future can best be seen in relation to the development of web services and the semantic web.

Web Services

Since the web has taken off as a medium for e-commerce and Business-2-Business (B2B) transactions, we are now seeing a move towards service-oriented applications, spurred on by major industry, large IT companies and academic research (Sollazzo *et al.*, 2002). In a service-oriented environment, web applications are specified in terms of the services they provide, the format and structure of the information transferred and the underlying protocols used in invoking these services. New standards and approaches are being developed to provide interoperability between processes and create a unified view of these services (Curbera *et al.*, 2001). These services are not only intended for human users, but also software agents working on behalf of a client, be it another agent or human user. Therefore agents are seen as a

critical component within the web service model and many of the developing standards and approaches have been based around existing agent research (McIlraith *et al.*, 2001).

Web services can be seen as the first step in the achieving the semantic web and moving towards a revolution in the way we interact with the internet. From a technical standpoint, a web service environment that can respond to high-level requests such as "*book me a flight to the next hypertext conference*", will need to analyse and understand the question, generate the appropriate queries, use some form of registry or broker to discover the necessary services and then actually invoke them, returning the results to the user. However, in amongst all of this activity there are also a set of constraints imposed on the agent, often implicitly, both by the user and the current context (McIlraith *et al.,* 2001). User profiling is essential if agents are to provide the correct level of service to the user. Personal preferences such as cost, class of travel (economy/business), dietary requirements etc., need to be satisfied alongside external constraints imposed by environment, or organisational bodies such as employers or governments. There are many situations where constraints can cause conflicts e.g. a user wants to fly business class but the employer is only willing to pay for economy class. Issues such as this need to be taken into account if users are to place any sort of trust in their agents; therefore, adapting to requirements is critical.

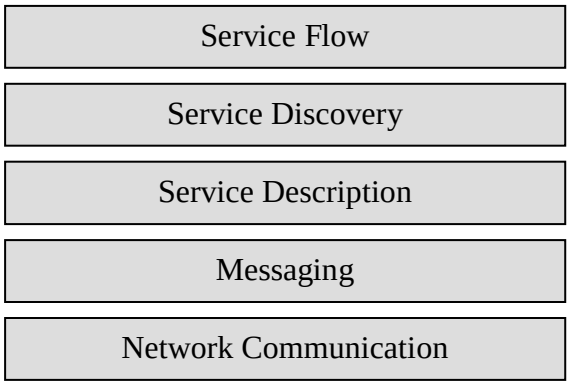

Figure 9-1. The Web Service Architecture

To exist within a web service architecture model, as presented in Figure 9-1, the agent-based adaptive framework has a big advantage because agent frameworks are already geared towards ontological knowledge representation and service discovery; tasks essential to web services. The agent framework fits in this environment by

providing an interface onto this web service world allowing users to interact with the various available services.

The interface agent would provide a natural language interface allowing the user to create queries and assign tasks to the AHS. The interface agent would convert the user request into a set of queries to the adaptation agent. The adaptation agent would query the user modeller agent for constraints applicable to the current query. After the query is modified by these constraints, the adaptation agent would locate the required web services through an external intermediary broker agent (such as a yellow pages service). Once the required services are identified, the adaptation agent would invoke the chosen service and return the results back to the user.

The web service model provides the next step for the evolution of web applications, however the long-term vision for this resource is in developing a *semantic web*, and here again agents and adaptive web-based systems play a crucial role.

Semantic Web

With the increased use of the web over the last decade, there have been attempts at making the web easier to use and more feature rich. Portals and search engines have taken the first step in attempting to manage the huge amount of information on the web. Search engines, like most human users of the web, rely on the structure and quality of the information on the web as a means of gauging relevant content and related material. Search engines will return thousands of hits to a single query, however relatively few will be of relevance to the user, and ultimately it is up to the user to find the information within the pages. One method of tackling this problem is to model users and record profiles in an attempt to improve the user's search queries and browsing experience. However this will only ever be a short term goal, a quick fix to a growing problem: the unstructured nature of the information on the web.

The semantic web is an idea developed by Web founder Tim Berners-Lee for a machine-understandable web (Berners-Lee, 1999). By supplying semantics and meta-data to the information that currently exists on the web, computers can be supplied with a level of understanding about the information they process. Recognising that the letters "IBM" do not just denote a string of characters, but refer to the name of a company,

and that a company has certain characteristics such as a share price, set of products, a CEO, a headquarters etc. would revolutionise the existing web experience. Search engines would be able to gain an understanding about the query, and respond, not only with a set of pages, but also with a set of services and related information. A query about a product could provide links to local shops selling the item, information about the company that owns the product, such as their share price, a feature comparison chart for the top models, or even a list of the best deals online. The semantic web would bring customers and companies closer, allow the inference of new information and provide new formats for the interchange of information between people, corporations and entire countries.

In such an environment, adaptive web-based systems become a large component of web interaction. The semantic web provides metadata and structure to information on a large scale, and applications that process this knowledge will do so based on the requirements of users. One of the problems associated with current client-side adaptive hypermedia systems is in moving away from the information domain, systems loose their association with the domain structure, thereby limiting the possible adaptive techniques offered. In the semantic web world, the structure and metadata attached to information is easily assessable and analysable by applications which allows them to personalised the content of this information. Structured information will encourage more applications to become adaptive and allow existing adaptive hypermedia systems to provide a much greater range of adaptive techniques and services then they currently offer.

The current proposed architecture of the semantic web features a multi-layer model resting on an underling XML data format with RDF, ontologies, logic, trust and proof constituting the higher levels. This architecture is shown in Figure 9-2.

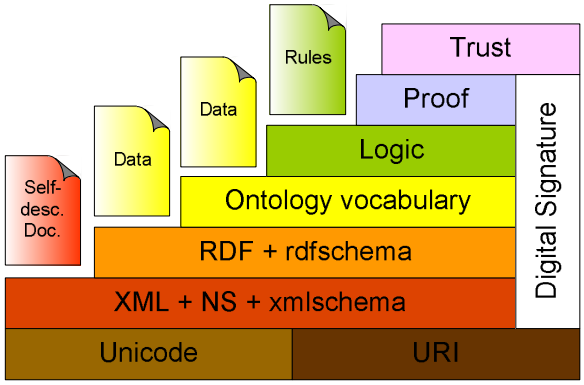

Figure 9-2. Tim Berners-Lee's Semantic Web cake

To complete this picture of the semantic web, agents are required to operate at the highest level of the model. The agents of the semantic web will formulate networks of trust between other agents, reason about existing knowledge, infer new information and use ontologies to facilitate this understanding. Agents have an essential role within the semantic web and this vision will ensure their development for the foreseeable future.

Whatever the future may bring, it is clear is that there is an increasing need for adaptation within programs. We have moved away from simple text interfaces and demand much more interaction from our applications. The need for personalisation, adaptable and adaptive web-based systems will continue to grow as the web expands. Agents that manipulate this information will hopefully provide the first step, and although the idea of real intelligent systems is still a distant dream, our continued and increasing interaction with digital devices will fuel the demand for further levels of adaptability for a long while to come.

Appendix A.

Table of Adaptive Hypermedia Application Architectures

SERVER-SIDE SYSTEMS	HYBRID SYSTEMS	CLIENT-SIDE SYSTEMS
ACE		
ADAPTS		
AHA!		
COOL Links		
CHEOPS		AiA
ELM-ART (I/II)		Antonmy
FireFly	WBI	Letizia
HYPADAPTER	WebWatcher	PAWS
IfWeb		Personal WebWatcher
INTERBOOK		WebMate
MetaLinks		
ProFusion		
PUSH		
SiteIF		

Appendix B.

Agent Frameworks

ActiveM3	Gypsy	Mobiware Middleware Toolkit
ADEPT	Hive	Mogent
ABE	iGENTM	MOLE
ABS	IMAJ	Mole
ADE	Infosleuth	MonJa
Agent Factory	Infospiders	MSA (ECRC)
Agent Tcl	Intelligent Agent Factory	muCode
AgentBuilder®	JACK Intelligent Agents	MAML
AgentSpace	JAE	MultiAgent Systems Tool
Agentx	JAFMAS	Nomadic Pict
Aglets	JAM	Nomads (Oasis)
Ajanta	James	Odyssey
Amase	JATLite	Open Agent Architecture
AMETAS	JAT	Open Sesame
Anchor Toolkit	JAFMAS	PACT
Anima	Java-2-go	Pathfinder
Ara	JavaNetAgents	Planet
ARCA	JavaSeal	Plangent
AuctionBot	JCAFE	ProcessLink
Bee-gent	JIAC	Reactive Action Package
Bond	J-SEAL2	Retsina
Bond Distributed Object System	Jumping Beans	rmi64
Cable	Kaariboga	SeMoA
CBorg	Kafka	SMASH
Concordia	KaOS	Social Interaction Framework
CyberAgents	Kasbah	SodaBot
D'Agents	Klaim	SoFAR
DASoc	KnowBots	SOMA
DECAF Agent Framework	Krest	Spheres of Commitment
Dejay	LALO	Subsumption architecture
DESIRE	LiveAgent	Swarm
DirectIA SDK	M0 Messengers	Telescript
dMARS	Magenta	Tierra
Echo	MAgNET	Traveler
Evolutionary Agent Societies (EAS)	MAP	TuCSoN
EXCALIBUR	MATS - Mobile agent teams	TuX
FarGo	Messengers	UMPRS
ffMAIN	Microsoft Agent	Versatile Intelligent Agents (VIA)
Firefly	Milenio	Voyager
GenA	MIPLACE	WASP
Gossip	MOA	Zeus
Grasshopper	Mobidget	

References

Amazon.com. (2002). *Amazon.com*. http://www.amazon.com, 2002.

Active Navigation. (2002). *Portal Maximizer*. http://www.activenavigation.com/PortalMax/, 2002.

Alatalo, T. & Peräaho, J. (2001). *Designing mobile-aware adaptive hypermedia*. Proceedings of the Third Workshop on Adaptive Hypertext and Hypermedia (AH '01), pp. 185-199, Jul, 2000.

Anderson, K. M., Taylor, R. N. & Whitehead, E. J. (1994). *Chimera: Hypertext for Heterogeneous Software Environments*. Proceedings of the ACM European conference on Hypermedia technology (ECHT '94), pp. 94-107, Sep, 1994.

Bailey, C. & Hall, W. (2000). *An Agent-Based approach to Adaptive Hypermedia Using a Link Service*. Proceedings of the First International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2000), pp. 260-263, Aug, 2000.

Bailey, C., El-Beltagy, S. & Hall, W. (2001). *Link Augmentation: A Context-Based Approach to Support Adaptive Hypermedia*. S. Reich, Tzagarakis, M. & De Bra, P. (Eds), Hypermedia: Openness, Structural Awareness, and Adaptivity, Springer, LNCS 2266, pp. 239-251, Aug, 2001.

Bailey, C., Hall, W., Millard, D. E. & Weal, M. J. (2002). *Towards Open Adaptive Hypermedia*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), Springer, LNCS 2347, pp. 36-46, May, 2002.

Balabanovic, M. & Shoham, Y. (1997). *Fab: Content-Based, Collaborative Recommendation*. Communications of the ACM, **40**(3), pp. 66-72, Mar, 1997.

Balabanovic, M. (1997). *An Adaptive Web Page Recommendation Service*. Proceedings of the First International Conference on Autonomous Agents, ACM Press, pp. 378-385, 1997.

Barnes & Noble. (2002). *Barnes & Noble.com*. http://www.bn.com, 2002.

Basu, C., Hirsh, H. & Cohen, W. (1998). *Recommendation as classification: Using social and content-based information in recommendation*. Proceedings of the Fifteenth National Conference on AI, AAAI Press, pp. 714-720, Jul, 1998.

Bates, J. (1994). *The role of emotion in believable agents*. Communications of the ACM, **37**(7), pp. 122-125, Jul, 1994.

Baudisch, P. & Brueckner, L. (2002). *TV Scout: Lowering the entry barrier to personalized TV program recommendation*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), Springer, LNCS 2347, pp. 58-68, May, 2002.

BBC. (2002). *BBC News*. http://news.bbc.co.uk, 2002.

Bellifemine, F., Rimassa, G. & Poggi, A. (1999). *JADE - A FIPA-Compliant Agent Framework*. Proceedings of the Forth International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents (PAAM '99), pp. 97-108, 1999.

Bergman, M., K. (2000). *The Deep Web: Surfacing Hidden Value*. A White Paper produced by BrightPlanet.com LLC, Jul, 2000.

Berners-Lee, T. (1999). *Weaving the Web*. Orion Publishing Group Ltd, ISBN: 0752820907, Nov, 1999.

Billsus, D., Brunk, C. A., Evans, C., Gladish, B. & Pazzani, M. (2002). *Adaptive Interfaces for Ubiquitous Web Access*. Communications of the ACM, **45**(5), pp. 34-38, May, 2002.

Billsus, D., Pazzani, M. J. & Chen, J. (2000). *A learning agent for wireless news access*. Proceedings of the International on Intelligent User Interfaces, pp. 33-36, 2000.

Bradley, K., Rafter, R. & Smyth, B. (2000). *Case-Based User Profiling for Content Personalisation*. Proceedings of the First International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2000), Springer, LNCS 1892, pp. 62-72, 2000.

Brusilovsky, P. & Eklund, J. (1998). *A Study of User Model Based Link Annotation in Educational Hypermedia*. Universal Computer Science, **4**(4), pp. 429-448, 1998.

Brusilovsky, P. (1996a). *Adaptive hypermedia: an attempt to analyse and generalize*. P. Brusilovsky, Kommers, P., and N. Streitz (Eds), Springer, 288-304, 1996.

Brusilovsky, P. (1996b). *Methods and Techniques of Adaptive Hypermedia*. User Modelling and User-Adaptive Interaction, **6**(2-3), pp. 87-129, 1996.

Brusilovsky, P. (2000). *Adaptive hypermedia: From intelligent tutoring systems to Web-based education*. G. Gauthier, Frasson, C. & VanLehn, K. (Eds), Proceedings of the Fifth International Conference on Intelligent Tutoring Systems (ITS 2000), Springer, LNCS 1839, pp. 1-7, Jun, 2000.

Brusilovsky, P. (2001). *Adaptive hypermedia*. A. Kobsa (Eds), User Modelling and User-Adapted Interaction, Ten Year Anniversary, 11, pp. 87-110, 2001.

Brusilovsky, P., Eklund, J. & Schwarz, E. (1998). *Web-based education for all: A tool for developing adaptive courseware*. Proceedings of the Seventh International World Wide Web Conference, pp. 291-300, Apr, 1998.

Brusilovsky, P., Schwarz, E., & Weber, G. (1996a). *ELM-ART: An intelligent tutoring system on World Wide Web*. C. Frasson, Gauthier, G., & Lesgold, A. (Eds), Intelligent Tutoring Systems, Springer, LNCS 1086, pp. 261-269, 1996.

Brusilovsky, P., Schwarz, E., & Weber, G. (1996b). *A tool for developing adaptive electronic textbooks on WWW*. Proceedings of the World Conference of the Web Society (WebNet'96), San Francisco, CA, USA, pp. 64-69, Oct, 1996.

Bush, V. (1945). *As we May Think*. The Atlantic Monthly, **176**(1), pp. 101-108, 1945.

Cannataro, M. & Pugliese, A. (2001). *XAHM: An XML-Based Adaptive Hypermedia Model and Its Implementation*. S. Reich, Tzagarakis, M. & De Bra, P. (Eds), Hypermedia: Openness, Structural Awareness, and Adaptivity, Springer, LNCS 2266, pp. 252-263, Aug, 2001.

Cannataro, M., Cuzzocrea, A., Mastroianni, C., Ortale, R. & Pugliese, A. (2002). *Modeling Adaptive Hypermedia with an Object-Oriented Approach and XML*. Proceedings of the Second International Workshop on Web Dynamics (WebDyn '02), Honolulu, Hawaii, May, 2002.

Carr, L., De Roure, D., Hall, W. & Hill, G. (1995). *The Distributed Link Service: A Tool for Publishers, Authors and Readers*. World Wide Web Journal, **1**(1), pp. 647-656, 1995.

Carr, L., Hall, W. & Hitchcock, S. (1998). *Link Services or Agent Services*. Proceedings of the Ninth ACM Conference on Hypertext, pp. 113-122, Jun, 1998.

Chen, L. & Sycara, K. (1998). *WebMate: Personal Agent for Browsing and Searching*. Proceedings of the Second International Conference on Autonomous Agents, ACM Press, pp. 132-139, May, 1998.

Church, K., W. & Hanks, P. (1989). *Word association norms, mutual information and lexicography*. Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics, pp. 76-83, 1989.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. & Sartin, M. (1999). *Combining Content-Based and Collaborative Filters in an Online Newspaper*. Proceedings of the ACM SIGIR Workshop on Recommender Systems, pp. 33-40, Aug, 1999.

Claypool, M., Le, P. & Wased, M. (2001). *Implicit interest indicators*. Proceedings of the ACM Intelligent User Interfaces Conference (IUI), 2001.

CNN. (2002). *CNN.com*. http://www.cnn.com, 2002.

Conklin, J. (1987). *Hypertext: An introduction and Survey*. IEEE Computer, **20**(9), pp. 17-41, 1987.

Cotter, P. & Smyth, B. (2000). *WAPing the web: content personalization for WAPenabled devices*. Proceedings of the First International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH2000), pp. 98-108, 2000.

Curbera, F., Nagy, W. & Weerawarana, S. (2001). *Web Services: Why and How*. Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '01), Oct, 2001.

Dale, J. & De Roure, D. (1997). *A Mobile Agent Architecture for Distributed Information Management*. Proceedings of the International Workshop on the Virtual Multicomputer, pp. 1-17, Mar, 1997.

Davis, H. C., Hall, W., Heath, I., Hill, G. & Wilkins, R. J. (1992). *Towards an Integrated Information Environment with Open Hypermedia Systems*. Proceedings of the ACM European conference on Hypermedia technology (ECHT '92), ACM Press, pp. 181-190, 1992.

Davis, H. C., Rizk, A., & Lewis, A. J. (1996). *OHP: A Draft Proposal for a Standard Open Hypermedia Protocol*. U. K. D. Wiil, S. (Eds), Proceedings of the Second Workshop on Open Hypermedia Systems, ACM Hypertext '96, pp. 27-53, Mar, 1996.

De Bra, P. & Calvi, L. (1998). *AHA: a Generic Adaptive Hypermedia System*. Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia (HYPERTEXT '98), Jun, 1998.

De Bra, P. (1999). *Design Issues in Adaptive Hypermedia Application Development*. Proceedings of the Second Workshop on Adaptive Systems and User Modeling on the World Wide Web, pp. 29-39, 1999.

De Bra, P., Aerts, A., Houben, G. J. & Wu, H. (2000). *Making General-Purpose Adaptive Hypermedia Work*. Proceedings of the WebNet Conference, pp. 117-123, 2000.

De Bra, P., Aerts, A., Smits, D. & Stash, N. (2002). *AHA! meets AHAM*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), Springer, LNCS 2347, pp. 381-384, May, 2002.

De Bra, P., Brusilovsky, P. & Houben, G. J. (1999a). *Adaptive Hypermedia, From Systems to Framework*. ACM Computing Surveys, **31**(4), pp. 1-6, Dec, 1999.

De Bra, P., Houben, G. J. & Wu, H. (1999b). *AHAM: A Dexter-based Reference Model for Adaptive Hypermedia*. Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 147-156, 1999.

De Roure, D., Hall, W., Reich, S., Pikrakis, A. Hill, G. & Stairmand, M. (1998). *An Open Architecture for Supporting Collaboration on the Web*. Proceedings of the Seventh International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '98), pp. 90-95, 1998.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W. & Harshman, R. A. (1990). *Indexing by Latent Semantic Analysis*. American Society of Information Science, **41**(6), pp. 391-407, 1990.

El-Beltagy, S. (2001). *An Agent Based Framework for Navigation Assistance and Information Finding in Context*. PhD thesis, University of Southampton, 2001.

El-Beltagy, S., De Roure, D. & Hall, W. (1999). *A Multiagent system for Navigation Assistance and Information Finding*. Proceedings of the Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp. 281-295, 1999.

El-Beltagy, S., De Roure, D. & Hall, W. (2000). *The Evolution of a Practical Agent-based Recommender System*. Proceedings of the Workshop on Agent-based Recommender Systems, Fourth International Conference on Autonomous Agents, Jun, 2000.

El-Beltagy, S., Hall, W., De Roure, D. & Carr, L. (2001). *Linking in Context*. Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia, pp. 151-160, Aug, 2001.

Engelbart, D. C. (1963). *A Conceptual Framework for the Augmentation of Man's Intellect*. **1**, Spartan Books, 1963.

Espinoza, F. & Höök, K. (1996). *A WWW Interface to an Adaptive Hypermedia System*. Proceedings of the First International Conference of the Practical Applications of Intelligent Agents and Multi-Agents (PAAM '96), pp. 143-167, Apr, 1996.

Ferrandino, S. Negro, A. & Scarano, V. (1997). *CHEOPS : Adaptive Hypermedia on World Wide Web*. Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Telecommunicazion Services (IDMS '97), Springer, LNCS 1309, Sep, 1997.

Finin, T., Labrou, Y. & Mayfield, J. (1997). *KQML as an Agent Communication Language*. J. Bradshaw (Eds), MIT Press, 291-316, 1997.

FIPA (2001). *The FIPA bake-off*. FIPA Inform! Newsletter, **2**(2), Apr, 2001.

FIPA. (2002). *The Foundation for Intelligent Physical Agents (FIPA)*. http://www.fipa.org/, 2002.

Flores-Mendez, R. A. (1999). *Towards the Standardization of Multi-Agent System Architectures: An Overview*. ACM Crossroads, Special Issue on Intelligent Agents, Association for Computer Machinery, (5.4), pp. 18-24, 1999.

Fountain, A., Hall, W., Heath, I. & Davis, H. C. (1990). *Microcosm: an open model with dynamic linking*. A. Rizk, Strietz, N. & Andre, J. (Eds), Proceedings of the European Conference on Hypertext (ECHT '90), pp. 298-311, Nov, 1990.

Franklin, S. & Graesser, A. (1997). *Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*. Proceedings of the ECAI '96 Workshop, Intelligent Agents III: Agent Theories, Architectures, and Languages (ATAL), Aug, 1997.

Gauch, S. & Futrelle, R. (1994). *Experiments in Automatic Word Class and Word Sense Identification for Information Retrieval*. Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR-94), pp. 425-434, Apr, 1994.

Giraffa, L. & Viccari, R. (1998). *The Use of Agents Techniques on Intelligent Tutoring Systems*. Proceedings of the Eighteenth International Conference of the Chilean Computer Science Society, Nov, 1998.

Google. (2002). *The Google Search Engine*. http://www.google.com, 2002.

Goose, S., Hall, W. & Reich, S. (2000). *Microcosm TNG: A Framework for Distributed Open Hypermedia*. IEEE MultiMedia, **7**(3), pp. 52-60, 2000.

Grønbæk, K. & Trigg, H. R. (1994). *Design issues for a Dexter-based hypermedia system*. Communications of the ACM, **37**(2), pp. 40-49, Feb, 1994.

Grønbæk, K., Hem, J. A., Madsen, O. L. & Sloth, L. (1993). *Designing Dexter-based cooperative hypermedia systems*. Proceedings of the Fifth ACM conference on Hypertext, Seattle, Washington, USA, ACM Press, pp. 25-38, 1993.

Gruber, T. R. (1993). *A translation approach to portable ontologies*. Knowledge Acquisition, **5**(2), pp. 199-220, 1993.

Grundin, J. (1994). *Groupware and Social Dynamics: Eight Challenges for Developers*. Communications of the ACM, **37**(1), pp. 92-105, 1994.

Guarino, N. & Giaretta, P. (1995). *Ontologies and Knowledge Bases: Towards a Terminological Clarification*. N. Mars (Eds), Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, IOS Press, pp. 25-32, 1995.

Guarino, N. (1998). *Formal Ontology and Information Systems*. N. Guarino (Eds), Proceedings of the Formal Ontology in Information Systems (FOIS '98), IOS Press, pp. 3-15, Jun, 1998.

Guilfoyle, C. & Warner, E. (1994). *Intelligent agents: The new revolution in software*. Ovum Report, 1994.

Halasz, F. G. & Schwartz, M. (1994). *The Dexter Hypertext Reference Model*. Communications of the ACM, **37**(2), pp. 30-39, Feb, 1994.

Hammond, N. V. & Allinson, L. (1989). *Extending hypertext for learning: an investigation of access and guidance tools*. A. M. Sutcliffe, L. (Eds), Proceedings of the Fifth conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and Computers V, Cambridge University Press, pp. 293-304, 1989.

Hardman, L., Bulterman, D. C. A. & van Rossum, G. (1994). *The Amsterdam hypermedia model: adding time and context to the Dexter model*. Communications of the ACM, **37**(2), pp. 50-62, Feb, 1994.

Henze, H.& Nejdl, W. (2002). *Knowledge Modeling for Open Adaptive Hypermedia*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), Springer, LNCS 2347, pp. 174-183, May, 2002.

Hewitt, C. (1977). *Viewing Control Structures as Patterns of Passing Messages*. Artificial Intelligence, **8**(3), pp. 323-364, 1977.

Hill, G. & Hall, W. (1994). *Extending the Microcosm Model to a Distributed Environment*. Proceedings of the ACM European conference on Hypermedia technology (ECHT '94), ACM Press, pp. 32-40, Sep, 1994.

Hill, G., Wilkins R. & Hall, W. (1993). *Open and Reconfigurable Hypermedia Systems: A Filter-Based Model*. Hypermedia, **5**(2), pp. 103-118.

Hohl, H., Böcker, H., & Gunzenhäuser, R. (1996). *HYPADAPTER: An Adaptive Hypertext System for Exploratory Learning and Programming*. User Modelling and User-Adapted Interaction, **6**(2-3), pp. 131-155, 1996.

Hothi, J. & Hall, W. (1998). *An Evaluation of Adapted Hypermedia Techniques using Static User Modelling*. Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia, pp. 45-50, Jun, 1998.

Hothi, J. (2001). *Using An Open Hypermedia System to Develop New Techniques in Adaptive Hypermedia*. PhD thesis, University of Southampton, 2001.

Huhns, M. N. & Singh, M. P. (1997). *Ontologies For Agents*. IEEE Internet Computing, **1**(6), pp. 81-83, 1997.

Janca, P. (1995). *Pragmatic Application of Information Agents*. BIS Strategic Decisions, Norwell, USA, May, 1995.

Jennings, N. R. (2000). *On Agent-Based Software Engineering*. Artificial Intelligence, **117**(2), pp. 277-296, 2000.

Jeon, H., Petrie, C. & Cutkosky, M. R. (2000). *JATLite: A Java Agent Infrastructure with Message Routing*. IEEE Internet Computing, **4**(2), pp. 87-96, Mar, 2000.

JESS. (2002). *Jess, the Expert System Shell for the Java Platform*. http://herzberg.ca.sandia.gov/jess/, 2002.

Joachims, T., Freitag, D. & Mitchell, T. (1997). *WebWatcher: A Tour Guide for the World Wide Web*. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI97), pp. 770-775, Aug, 1997.

Kappel, G., Rausch-Schott, S., Reich, S., Retschitzegger, W. (1996). *Benefits of Open Hypermedia Systems Using Advanced Database Concepts*. Proceedings of the GI/OCG Annual Computer Science Conference, Informatik '96, pp. 179-195, Sep, 1996.

Kappel, G., Retschitzegger, W. & Schwinger, W. (2000). *Modeling Customizable Web Applications - A Requirement's Perspective*. Proceedings of the International Conference on Digital Libraries (ICDL 2000), pp. 387-399, Nov, 2000.

Kay, J., Kummerfeld, B. & Lauder, P. (2002). *Personis: A server for User Models*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), Springer, LNCS 2347, pp. 203-212, May, 2002.

Kobsa, A. (1993). *User Modeling: Recent Work, Prospects and Hazards*. M. Schneider-Hufschmidt, Kühme, T. & Malinowski, U. (Eds), Adaptive User Interfaces: Principles and Practise. North-Holland, 111-128, 1993.

Koch, N. & Kraus, A. (2002). *The expressive Power of UML-based Web Engineering*. D. Schwabe, Pastor, O., Rossi, G. & Olsina, L. (Eds), Proceedings of the Second International Workshop on Web-oriented Software Technology (IWWOST´02), Jun, 2002.

Koch, N. & Wirsing, M. (2001). *Software Engineering for Adaptive Hypermedia Systems*. Proceedings of the Third Workshop on Adaptive Hypertext and Hypermedia, pp. 205-210, Jul, 2001.

Koch, N. & Wirsing, M. (2002). *The Munich Reference Model for Adaptive Hypermedia Applications*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), Springer, LNCS 2347, pp. 213-222, May, 2002.

KQML. (2002). *The Knowledge Query and Manipulation Language (KQML)*. http://www.cs.umbc.edu/kqml/, 2002.

Langley, P., Iba, W. & Thompson, K. (1992). *An Analysis of Bayesian Classifiers*. Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), San Mateo, CA, USA, AAAI Press/MIT Press, pp. 223-228, 1992.

Lieberman, H. (1995). *Letizia: An agent that assists Web browsing*. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp. 924-929, 1995.

Lowe, D. & Hall, W. (1999). *Hypermedia and the Web: an Engineering Approach*. Wiley, ISBN: 0471983128, 1999.

Lycos. (2002). *The Lycos Search Engine*. http://www.lycos.com, 2002.

Maes, P. (1994). *Agents that Reduce Work and Information Overload*. Communications of the ACM, **37**(7), pp. 31-40, Jul, 1994.

Maglio, P., P. & Farrell, S. (2000). *LiveInfo: Adapting web experience by customization and annotation*. Proceedings of the First International Conference on Adaptive

Hypermedia and Adaptive Web-based Systems (AH2000), Springer, LNCS 1892, pp. 144-154, Aug, 2000.

Maltz, D. & Ehrlich, E. (1995). *Pointing the way: Active collaborative filtering*. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '95), pp. 202-209, May, 1995.

MASIF. (2002). *The Mobile Agent System Interoperability Facility (MASIF)*. http://www.fokus.gmd.de/research/cc/ecco/masif/, 2002.

McCarthy, J. (1959). *Programs with Common Sense*. Proceedings of the Teddington Conference on the Mechanization of Thought Processes, pp. 77-84, Dec, 1959.

McIlraith, S., Son, T. C. & Zeng, H. (2001). *Semantic Web Services*. IEEE Intelligent Systems, Special Issue on the Semantic Web, **16**(2), pp. 46-53, Mar, 2001.

Michaelides, D. T., Millard, D. E., Weal, M. J. & De Roure, D. C. (2001). *Auld Leaky: A Contextual Open Hypermedia Link Server*. Proceedings of the Seventh Workshop on Open Hypermedia Systems (Hypertext '01), Springer, LNCS 2266, pp. 52-64, Aug, 2001.

Millard, D. (2000). *Hypermedia Interoperability: Navigating the Information Continuum*. PhD thesis, University of Southampton, Dec, 2000.

Milojicic, D., Breugst, M., Busse, I., Campbell, J., Covaci, S., Friedman, B., Kosaka, K., Lange, D., Ono, K., Oshima, M., Tham, C., Virdhagriswaran, S., & White, J. (1998). *MASIF The OMG Mobile Agent System Interoperability Facility*. Proceedings of the International Workshop on Mobile Agents (MA '98), Springer, LNCS 1477, pp. 14-15, Sep, 1998.

Minar, N. (1998). *Designing an Ecology of Distributed Agents*. Masters thesis, Massachusetts Institute of Technology, 1998.

Mladenic, D. (1996). *Personal WebWatcher: Implementation and Design*. Report Number: IJS-DP-7472, Department of Intelligent Systems, J.Stefan Institute, Slovenia, 1996.

Mladenic, D. (1999). *Text-Learning and Related Intelligent Agents: A Survey*. IEEE Intelligent Systems, **14**(4), pp. 44-54, 1999.

Mooney, R., J. & Roy, L. (2000). *Content-Based Book Recommending Using Learning for Text Categorization*. Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 195-204, Jun, 2000.

Moore, G. E. (1965). *Cramming More Components Onto Integrated Circuits*. Electronics, **38**(8), pp. 114-117, Apr, 1965.

Moreau, L., Gibbins, N., De Roure, D., El-Beltagy, S., Hall, W., Hughes, G., Joyce, D., Kim, S., Michaelides, D., Millard, D., Reich, S., Tansley, R. & Weal, W. (2000). *SoFAR with DIM Agents: An Agent Framework for Distributed Information Management*. Proceedings of the Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents (PAAM '00), pp. 369-388, Apr, 2000.

Moreau, L., Tan, V. & Gibbins, N. (2001). *Transparent Migration of Mobile Agents.* IEE Seminar: Mobile Agents - Where are They Going? Apr, 2001.

Morita, M. & Shinoda, Y. (1994). *Information filtering based on user behavior analysis and best match text retrieval*. Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, pp. 272-281, Jul, 1994.

Moukas, A. (1996). *Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem*. Proceedings of the First International Conference of the Practical Applications of Intelligent Agents and Multi-Agents (PAAM '96), pp. 437-457.

Murray, T. (1999). *Authoring Intelligent Tutoring Systems: An analysis of the state of the art*. Artificial Intelligence in Education, **10**, pp. 98-129, 1999.

Murray, T. (2002). *MetaLinks: Authoring and Affordances for Conceptual and Narrative Flow in Adaptive Hyperbooks*. Artificial Intelligence in Education, **13**, 2002.

Nelson, T. H. (1965). *A File Structure for the Complex, the Changing, and the Indeterminate*. Proceedings of the 20th ACM National Conference, pp. 84-100, Aug, 1965.

Newell, A., J., Shaw, C. & Simon, H. A. (1960). *Report on a general problem solving program*. Proceedings of the International Conference on Information Processing (UNESCO), pp. 256-64, 1960.

Ng, M., Hall, W., Maier, P. & Armstrong, R. (2002). *Using Effective Reading Speed to Integrate Adaptivity into Web-based Learning*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), Springer, LNCS 2347, pp. 428-431, May, 2002.

Nielsen, J. & Lyngbaek, U. (1990). *Two field studies of hypermedia usability*. R. McAleese, & Green, C. (Eds), Hypertext: State of the Art, Ablex, pp. 64-72, 1990.

Niemeyer, P. & Peck, J. (1997). *Exploring Java*. 2nd Edition, O'Reilly, ISBN: 1-56592-271-9, 1997.

NUA Internet Surveys. (2002). *How Many Online*. http://www.nua.ie/surveys/how_many_online/world.html, 2002.

Nwana, H. (1996). *Software Agents: An Overview*. Knowledge Engineering Review, **11**(3), pp. 1-40, Nov, 1996.

O'Brien, P. & Nicol, R. (1999). *FIPA - Towards a Standard for Software Agents*. BT Technology, **16**(3), pp. 51-59, 1999.

O'Hare, G., Sewell, K., Murphy, A. & Delahunty, T. (2000). *ECHOES: An Immersive Training Experience*. Proceedings of the First International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2000), pp. 179-188, Aug, 2000.

Oppermann, R. & Specht, M. (1999). *A Nomadic Information System for Adaptive Exhibition Guidance*. Proceedings of the International Cultural Heritage Informatics Meeting (ICHIM '99), pp. 103-109, 1999.

Pazzani, M. J. & Billsus, D. (1999). *Adaptive web site agents*. Proceedings of the Third International Conference on Autonomous Agents, pp. 394-395, 1999.

Pearl, A. (1989). *Sun's link service: A protocol for linking*. Proceedings of the First Hypertext Conference (Hypertext '89), ACM Press, pp. 137-146, 1989.

Pennock, D., M., Horvitz, E. & Giles, C., L. (2000). *Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering*. Proceedings of the Seventeenth National Conference on Artificial Intelligence, pp. 729-734, Jul, 2000.

Petrie, C., J. (2000). *Agent-Based Software Engineering*. P. W. Ciancarini, M. (Eds), Agent-Oriented Software Engineering, Springer, LNAI 1957, pp. 58-76, 2000.

Pikrakis, A., Bitsikas, T., Sfakianakis, S., Hatzopoulos, M., De Roure, D., Hall, W., Reich, S., Hill, G. & Stairmand, M. (1998). *MEMOIR - Software Agents for Finding Similar Users by Trails*. H. S. N. Nwana, D. T. (Eds), Proceedings of the Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents (PAAM '98), pp. 453-466, Mar, 1998.

Porter, M. F. (1980). *An algorithm for suffix stripping*. Program, **14**(3), pp. 130-137, Jul, 1980.

Rao, A. S. & Georgeff, M. P. (1991). *Modeling rational agents within a BDI-architecture*. J. Allen, Fikes, R. & Sandewall, E. (Eds), Proceedings of the

Second International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, pp. 473-484, 1991.

Rich, E. (1979). *User Modeling via Stereotypes*. Cognitive Science, **3**(4), pp. 329-354, 1979.

Rocha, L., M. (2001). *Adaptive Recommendation and Open-Ended Semiosis.* Kybernetes, **30**(5-6), 2001.

Salton, G. (1991). *Developments in automatic text retrieval*. Science, **253**, pp. 974-980, Aug, 1991.

Schafer, J. B., Konstan, J., & Riedl, J. (1999). *Recommender Systems in E-Commerce*. Proceedings of the ACM E-Commerce conference, pp. 158-166, Nov, 1999.

Schoech, V., Specht, M. & Weber, G. (1998). *ADI An Empirical Evaluation of a Pedagogical Agent*. Proceedings of the World Conference on Educational Multimedia (ED-MEDIA '98), pp. 380-406, 1998.

Searle, J. (1969). *Speech Acts*. Cambridge University Press, ISBN: 052109626X, Jan, 1969.

Silveira, R. A. & Vicari, R. M. *Improving Interactivity in e-Learning Systems with Multi-Agent Architecture*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), pp. 466-471, May, 2002.

Sinclair, P., Martinez, K., Millard, D. & Weal, M. (2002). *Links in the Palm of your Hand: Tangible Hypermedia using Augmented Reality*. Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia, pp. 127-136, Mar, 2002.

Sleeman, D. H. & Brown, J. S. (1982). *Intelligent Tutoring Systems.* London, Academic Press, ISBN: 0126486808, 1982.

Smart, T. (1997). *The Oxford Paperback Dictionary and Thesaurus*. Oxford University Press, ISBN: 0198613318, 1997.

Sollazzo, T., Handschuh, S., Staab, S. & Frank, M. (2002). *Semantic Web Service Architecture - Evolving Web Service Standards toward the Semantic Web*. Proceedings of the Fifteenth International FLAIRS Conference, AAAI Press, May, 2002.

Specht, M. & Oppermann, R. (1998). *ACE - Adaptive Courseware Environment*. The New Review of Hypermedia and Multimedia, **4**, pp. 141-161, 1998.

Specht, M., Weber, G., Heitmeyer, S. & Schöch, V. (1997). *AST: An adaptive WWW-Courseware for Statistics*. Proceedings of the Workshop on Adaptive Systems and User Modeling on the WWW (UM '97), pp. 91-96, Jun, 1997.

Sunderam, A. V. (2002). *Automated Personalization of Internet News*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH '02), Springer, LNCS 2347, pp. 348-357, May, 2002.

Turing, A., M. (1950). *Computing Machinery and Intelligence*. Mind, (59), pp. 433-460, Oct, 1950.

Urban-Lurain, M. (1996). *Intelligent Tutoring Systems: An Historic Review in the Context of the Development of Artificial Intelligence and Educational Psychology*. http://aral.cse.msu.edu/Publications/ITS/its.htm, 1996.

Uschold, M. & Gruninger, M. (1996). *Ontologies: Principles, Methods and Applications*. Knowledge Engineering Review, **11**(2), pp. 93-155, 1996.

Venezky, R. & Osin, L. (1991). *The intelligent design of computer-assisted instruction*. New York, Longman Publishing Group, ISBN: 0801303907, 1991.

Vitaglione, G., Quarta, F. & Cortese, E. (2002). *Scalability and Performance of JADE Message Transport System*. Proceedings of the Challenges in Open Agent Systems Workshop in the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS '02), Jul, 2002.

W3C. (2002). *XML Linking Language (XLink) Version 1.0*. http://www.w3.org/TR/xlink/, 2002.

Wagner, D. N. (2000). *Liberal Order for Software Agents? An economic analysis*. Journal of Artificial Societies and Social Simulation, **3**(1), 2000.

Wantz, L. J. & Miller, M. (1997). *Toward User-Centric Navigation of the Web: COOL Links using SPI*. Proceedings of the Sixth International World Wide Web Conference, Apr, 1997.

Watson, A. & Sasse, M. A. (1998). *Measuring Perceived Quality of Speech and Video in Multimedia Conferencing Applications*. Proceedings of the ACM Multimedia Conference, pp. 55-60, Sep, 1998.

Weal, M. J., Millard, D. E., Michaelides, D. T. & De Roure, D. C. (2001). *Building Narrative Structures Using Context Based Linking*. Proceedings of the Twelfth ACM conference on Hypertext (Hypertext '01), pp. 37-38, 2001.

Weber, G. & Brusilovsky, P. (2001). *ELM-ART: An adaptive versatile system for Web-based instruction*. Artificial Intelligence in Education, Special Issue on Adaptive and Intelligent Web-based Educational Systems, **12**(4), 2001.

Weber, G. & Möllenberg, A. (1994). *ELM-PE: A knowledge-based programming environment for learning LISP*. Proceedings of the World Conference on Educational Multimedia (ED-MEDIA '94), pp. 557-562, 1994.

Weber, G. & Specht, M. (1997). *User Modelling and Adaptive Navigation Supporting WWW-based Tutoring Systems*. Proceedings of the Sixth International Conference on User Modeling, pp. 289-300, Jun, 1997.

Weber, G., Kuhl, H. C. & Weibelzahl, S. (2001). *Developing adaptive internet based courses with the authoring system NetCoach*. S. Reich, Tzagarakis, M. & De Bra, P. (Eds), Hypermedia: Openness, Structural Awareness, and Adaptivity, Springer, LNCS 2266, pp. 226-238, 2001.

Wiil, U. K., & Whitehead, E. J. (1997). *Interoperability and Open Hypermedia Systems*. Proceedings of the Third Workshop on Open Hypermedia Systems, pp. 137-145, Apr, 1997.

Wooldridge, M. & Jennings, N., R. (1995). *Intelligent Agents: Theory and Practice*. The Knowledge Engineering Review, **10**(12), pp. 115-152, 1995.

Wu, H., De Kort, E. & De Bra, P. (2001). *Design Issues for General-Purpose Adaptive Hypermedia Systems*. Proceedings of the ACM Conference on Hypertext and Hypermedia (Hypertext '01), pp. 141-150, Aug, 2001.

Wu, H. (2002). *A Reference Architecture for Adaptive Hypermedia Applications*. PhD, Techniche Universiteit Eindhoven, Nov, 2002.

Yankelovich, N., Haan, B. J., Meyrowitz, N. & Drucker, S. M. (1988). *Intermedia: The Concept and the Construction of a Seamless Information Environment*. IEEE Computer, **21**(1), pp. 81-96, 1988.