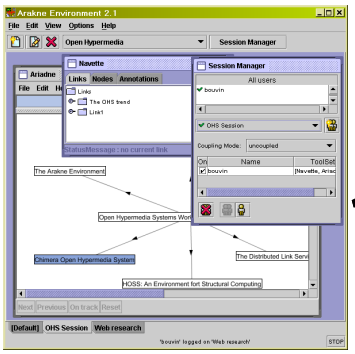
Augmenting the Web through Open Hypermedia

The Development of the Arakne Environment, a Collaborative Open Hypermedia System for Web Augmentation



Niels Olof Bouvin

Ph.D. Thesis

University of Aarhus

November 2000

Contents

List of Figures

List of Tables

Chapter 1

Foreword

1.1 Structure of this Thesis

This report forms the first part of of my Ph.D. thesis titled “Augmenting the Web through Open Hypermedia”. This part summarises my work and relate it to the hypermedia and Web research fields. The second part is formed by the five papers described below in Section 1.2, all dealing with augmenting the Web with open hypermedia or collaboration. The third and final part of my Ph.D. is the Arakne Environment, a system for creating open hypermedia structures on top of Web pages, which availability is described in Appendix B. It is also available on the CD delivered with this text.

Foreword This chapter

Danish Summary A summary of the thesis in Danish.

Introduction and Motivation The Web is briefly introduced, along with some scenarios, highlighting some shortcomings of the Web. These scenarios are used to introduce the term ’Web augmentation’, that is using open hypermedia technology to structure the Web.

Hypermedia and the Web My work is related to both open hypermedia research and the Web community. This chapter gives an overview of both fields as well as a brief introduction to the relevant CSCW topics, with special focus on the areas directly related to my work.

Related work There are a growing number of commercial products aimed at Web augmentation. This chapter introduces some of these systems along with a description of XLink, a W3C XML standard for navigational hypermedia.

Contributions My contributions to the field based on the papers described in Section 1.2. I have mainly worked directly on Web augmentation and this is reflected in the papers. My experiences with the Open Hypermedia Systems Working Group compliant Construct servers are reported, as is work on extending and generalising the approach to shared awareness found in the Arakne Environment.

Challenges for Web Augmentation While Web augmentation is greatly eased by the accessibility of Web standards, there are some lessons with Web browsers and Java accumulated through my work, that identifies some of the obstacles remaining.

Conclusion This chapter summaries the results, I have achieved, and sets the directions of my future work. The Arakne Environment and its related technologies have now reached a point where work can begin in earnest.

Use studies Two cases, one detailing the need for Web augmentation at Landbrugets Rådgivningscenter, another showing Web augmentation in the form of guided tours used in practice at the Opasia Web portal (the largest Danish ISP).

Installing the Arakne Environment A short description detailing how to download and install the Arakne Environment.

Vocabulary Short definitions of various terms used throughout this text.

1.2 Papers

The core part of this thesis consists of the following papers, included with the thesis:

- N. O. Bouvin. Unifying strategies for Web augmentation. In *Proceedings of the* 10^{th} *ACM Hypertext Conference*, pages 91–100, Darmstadt, Germany, Feb. 1999. This paper introduces the term "Web augmentation". Based on a study of a number of Web augmentation tools, a general framework to model Web augmentation, as well as an implementation using this framework is presented. Referred to in this text as [17] and then marked with [P1].

- N. O. Bouvin and R. Schade. Integrating temporal media and open hypermedia on the World Wide Web. In *Computer Networks — The International Journal of Computer and Telecommunications Networking*, (31):1453–1465, 1999. Traditionally, Web augmentation has concentrated on adding links to Web pages, that is, HTML documents. This paper describes how temporal media, such as video or audio clips, may be augmented with links on the Web. Referred to in this text as [20] and then marked with [P2].

- N. O. Bouvin. Experiences with OHP and issues for the future. in *Lecture Notes in Computer Science* 1903. Springer, 2000. Based on the work of porting the Arakne Environment to the Open Hypermedia Protocol, this paper reflects on the state of the standards developed by the Open Hypermedia Systems Working Group, and discusses how this work can continue in the future. Referred to in this text as [19] and then marked with [P3].

- N. O. Bouvin. Collaborative Web-based open hypermedia and mutual awareness. To be submitted for publication. This paper describes how collaborative work and shared awareness is supported in the Arakne Environment, with special focus on how collaboration can be seamlessly integrated into the interaction with the system. Referred to in this text as [15] and then marked with [P4].

- K. M. Anderson and N. O. Bouvin. Enabling project awareness and intersubjectivity via hypermedia-enabled event trails. Submitted for publication. This paper introduces the iScent architecture, a system to support shared awareness and intersubjectivity between co-workers. As such it is a continuation and generalisation of the awareness tools already present in the Arakne Environment. Referred to in this text as [6] and then marked with [P5].

1.3 The Arakne Environment

The Arakne Environment is a tangible result of my Ph.D. It is a collaborative open hypermedia system aimed at Web augmentation. The basis of the system is an environment wherein an open set of hypermedia tools, known as "views", exists and currently provide the user with navigational, spatial, and guided tour structuring mechanisms for Web pages. Details on how to obtain and install the system is found in Appendix B.

http://www.bouvin.net/Arakne/

It had not reached its current state without the help of the following Coconut student programmers, to whom I am grateful: René Thomsen, Michael Bang Nielsen, and Henning Jehøj Madsen.

1.4 My Background

It was 1993, and I was a summer student at the Department for Electronics and Computing for Physics in Geneva CERN, when Tim Berners-Lee introduced me to the Web... In *those* days, you could still find Web pages (at CERN and at NCSA) that listed the whole of the Web... Since then things have changed considerably. In spring 1994 I decided to attend a new class on Hypermedia, because I thought it sounded interesting (it was). Four concepts particularly caught my attention at that time — Bush's trail blazer, Engelbart's ambition to augment the human intellect, Nelson's Docuverse, and the elegance of NoteCards.

http://www.cern.ch/

http://www.ncsa.edu/

I received my Master's Degree in late July 1996 (mastering in computer science with a minor in mathematics) at the Department of Computer Science, University of Aarhus, Denmark. Together with Christina Nielsen and Christian Mulvad Sejersen I had been involved in an activity theoretical HCI evaluation of a research prototype created by EuroPARC Cambridge for the Esprit EuroCODE project, which we documented in our Master's thesis "Spirits in a Material World", with Susanne Bødker as our supervisor. Our contribution to the EuroCODE project was published in the EuroCODE Deliverable 1.4.2.

After graduation I accepted an offer to work at the Department of Computer Science, where I developed the DHM/WWW applet, the first application to offer client-side open hypermedia links in Web pages by integrating an ordinary Web browser. The results were published in [48], co-authored by Kaj Grønbæk, Lennert Sloth, and myself. This paper later won the Engelbart Award for best paper at Hypertext 1997 in Southampton, England.

This wetted my appetite for what I later would term "Web augmentation", and in September 1997 I was accepted as a Ph.D. student at Aarhus with Kaj Grønbæk as my supervisor, with the topic of open hypermedia on the Web. I became a member of the Coconut project, a project collaboration between the Department of Computer Science and Tele-Danmark Internet (the largest Danish ISP). The aim of the Coconut project was twofold: to make new hypermedia research, and explore the possibilities for collaborative component-based Web technology. The project ended by the end of 1999, while my Ph.D. studies continued to the end of August 2000.

http://www.opasia.dk/

It has been exciting to work within Coconut — pure research coupled with the needs of a corporate entity — and I wish to thank all the participants: Kaj Grønbæk, Ole Hedegård, Jesper Jühne (original co-developer of Ariadne), Henning Jehøj Madsen, Michael Bang Nielsen, Peter J. Nürnberg, Lars Pind, Olav Reinert (original co-developer of CAOS), René Schade (original developer of Mimicry), Jörg Schneider, Kristine Thomsen, René Thomsen, and Uffe Kock Wiil

I spent the first five months of 2000 at the University of Colorado, Boulder, visiting Kenneth M. Anderson. The stay was very enjoyable and fruitful with good discussions and work on hypermedia and with the development of the iScent framework (Section 6.5) as the main result.

1.5 Acknowledgements

A very special thank you to the people, without whose comments and suggestions, this text would look quite different: Kaj Grønbæk, Christina Nielsen, and Marianne Graves Petersen.

Chapter 2

Danish Summary

Jeg har i min Ph.D. beskæftiget mig med, hvorledes man udbygger funktionaliteten af World Wide Web ved hjælp af åben hypermedieteknologi. WWW er en stor success, men visse ting er fortsat vanskelige. Man kan således ikke lave links fra andre folks sider, da linking indebærer modifikation af den Web side, som man linker fra. De links, man *kan* skabe fra egne sider er simple unidirektionale links med ét endepunkt. Som struktureringsmekanisme er et så simpelt link begreb relativt begrænset. En anden begrænsning af WWW er den ringe understøttelse for samarbejde.

Problemet med struktur på WWW vil ikke blive mindre med tiden, idet WWW fortsat vokser med stor hast. F.eks. gør en lang række danske statslige institutioner i stigende grad deres materiale tilgængeligt over WWW. Hertil kommer de daglige referencer til Web sider for uddybende information i medierne.

`http://www.danmark.dk/`

Hypermedieforskningen har i løbet af de sidste 55 år produceret en mængde spændende systemer med struktureringsmekanismer af meget forskellig art. Fra traditionelt navigationelt hypermedie (dvs. links og ankre), som også anvendes på WWW, til guidede ture, spatialt eller taksonomisk hypermedie. Hertil kommer, at hypermedie systemer allerede tidligt fokuserede på samarbejdsstøtte til specielt forfattervirksomhed af den ene eller anden art.

De klassiske hypermedie systemer var lukkede eller monolitiske. Dette vil sige, at de kun i ringe grad eller slet ikke var i stand til at håndtere dokumenter lavet med andre værktøjer. Såfremt man ønskede at benytte hypermedie, måtte man således forlade sine eksisterende programmer, og forlade sig på de i hypermedie systemet tilgængelige. Dette medførte en begrænset udbredelse af hypermedie teknologien. I 1989 publicerede Meyrowitz sin artikel “The Missing Link: Why We All doing Hypertext Wrong” [78], hvor han identificerede den manglende støtte for almindeligt forekommende værktøjer (såsom tekstbehandlingsprogrammer eller regneark) som hypermedie forskningens største problem. Denne artikel afstedkom en fokusering på at integrere hypermediefunktionalitet ind i almindelige programmer. Dette område betegnes sædvanligvis som åbent hypermedie forskning. Området er i år 2000 elleve år gammelt (regnet efter den første publikation, der omhandlede et åbent hypermediesystem, Sun’s Link Service [86]), og i den periode er der blevet udviklet mange forskellige åbne hypermediesystemer, der har demonstreret, at det er muligt at tilbyde hypermediefunktionalitet i programmer, der ikke er specielt udviklede til formålet.

Et andet område, der også i år er elleve år gammelt er World Wide Web. Det blev oprindeligt skabt af Tim Berners-Lee på kerneforskningsinstituttet CERN i Geneva i Schweiz. WWW er reelt et monolitisk hypermediesystem, der primært fungerer ved

hjælp af dokument formatet HTML. Ironisk nok er WWW det eneste kendte eksempel på et monolitisk hypermediesystem, der har formået at få endog stor udbredelse, og som oven i købet har evnet at få andre systemer til at tilpasse sig. Den store succes, som WWW har nydt, må hovedsageligt tilskrives den simple bagvedliggende arkitektur. Denne arkitektur har imidlertid også, som nævnt i indledningen, nogle begrænsninger.

Mit arbejde har taget udgangspunkt i den åbne hypermedieforskning med det formål at udvide hypermedie funktionaliteten af WWW. Dette mål kalder jeg "Web augmentation", hvilket løseligt kan oversættes til "Web forbedring". I forbindelse med mit arbejde har jeg dels udviklet en række prototyper, der demonstrerer åbnet hypermedie kombineret med WWW, dels arbejdet med den generelle arkitektur for sådanne værktøjer, og endeligt beskæftiget mig med, hvorledes samarbejde kan understøttes bedre på nettet.

Det foreløbige højdepunkt af mit arbejde er "the Arakne Environment" (kort Arakne). Arakne er en hypermedieomgivelse, der i øjeblikket understøtter navigationelt, guidet tur og spatialt hypermedie på WWW. Ved hjælp af denne omgivelse kan en bruger skabe noter og tovejs mange til mange links på kryds og tværs af WWW, desuagtet om vedkommende har skriveadgang til de pågældende sider eller ej. Systemet gemmer de skabte hypermediestrukturer i en hypermedie server, og først når en Web side hentes af brugeren, indsættes de relevante links og noter, således at de fremstår på skærmen. Såfremt flere brugere er tilknyttet de samme hypermedieservere, kan de igennem Arakne samarbejde omkring skabelsen af hypermediestrukturer. Hvis en bruger måtte ønske det, er det også muligt at eksportere hypermediestrukturer til en XML fil, som brugeren så kan sende til andre via elektronisk post, eller placere på sin Web side. Arakne er et Windows program og anvender Microsoft Internet Explorer som sin Web browser.

I forbindelse med udviklingen af Arakne og de systemer, der gik forud, har jeg naturligvis haft lejlighed til at høste en del erfaringer med kombinationen af åbne hypermedie systemer og WWW. Disse erfaringer (såsom problemerne omkring at skulle tilpasse en Web browser til denne anvendelse) er beskrevet i dette dokument.

Foruden selve udviklingen af diverse systemer, består min Ph.D. også af fem artikler, som er vedlagt dette dokument. Disse artikler beskriver udviklingsprocessen, der førte til Arakne, herunder det generelle Arakne Framework; arbejdet med at udvide Arakne, således at man kan linke til og fra videofilm på WWW; tilpasningen af Arakne til Construct hypermedieserverne; samarbejdsstøtte i Arakne; og endeligt en generel arkitektur til at kunne understøtte intersubjektivitet på Internet skala.

Mit arbejde med "Web augmentation" har vist, dels teoretisk og praktisk, at WWW med fordel kan udvides med mere avancerede hypermediefunktionalitet. Den udvidelse har til formål at forbedre WWW, men ikke at erstatte det. WWW er i dag så stort, at en udskiftning er aldeles urealistisk. Hertil kommer, at WWW på mange andre områder end netop hypermediemodellen er et glimrende system, hvilket dets store udbredelse også vidner om. Snarere er det interessant at se på de anvendelsesområder, hvor WWW har svagheder, og afhjælpe disse, og mit arbejde viser, at dette i høj grad er muligt.

Chapter 3

Introduction and Motivation

The Web as it stands today is a tremendous success. It is the single largest collection of information ever created, and it is readily available to everyone with access to a Web browser. As hypermedia systems go, the Web is unfortunately lacking in some respects, especially with regards to the hypermedia model and the support for collaborative work. I will in this introduction illustrate some of the shortcomings of the Web through some scenarios.

3.1 Working on the Web

> John D. Hansen is an agricultural consultant, and he spends most of his time researching and writing up reports and articles for the farming community. His company has in the last couple of years started to focus on the Web and John is currently working on the Web version of his most recent article about herbicides. John is not quite satisfied with the contents of the Web pages from one large herbicide manufacturer, and decides to put in a link on the manufacturer's Web page to his own analysis. Later, when John publishes his article, users of his company's site (and users of their proxy) can see a link to his report on the manufacturer's Web site. While he is at it, John creates a guided tour juxtaposing some recent video clips from national TV of sensationalistic claims with regards to the use of the herbicide "Ground up" with the actual test reports and results, adding his own comments.

This scenario illustrates several important points, much of which cannot be realised with the Web, as it stands today. The Web is essentially (baring self publishing) a read-only media, where users may access all kinds of information, but are barred from making annotations, or modifying (their view of) documents. The study and development of techniques to accomplish this using open hypermedia has been the focus of my Ph.D.

An area, where the Web and Web browsers so far have been severely lacking is the support for critical reading. There may be a lot of information available, but users are not free to annotate these pages as they can with a book [73]. A recurring theme on the topic of the Internet and the Web is the creation of special interest groups and NGOs (non-goverment organisations), that span geographical borders. Related is the

proliferation of meta Web sites, that largely consists of references (with short introductions) to pages on other Web sites. While still young, there is also such a thing as Web journalism, where journalists review and comment on Web sites, or create overviews of specific topics. Common to these cases is that the creators of comments are limited, when referring to other Web pages: they can either make whole page references, or copy and paste relevant parts into their own documents and thus leave it as an exercise to the reader to locate the quoted portion in the original Web page.

> Consider another situation, this time within an organisation. Many organisations deploy intranet with Web servers accessible only from the company network. If several groups within a project depend on the same technical specification, a Web server is well suited for making these documents easily accessible. However, the Web is not only documents, it is also links. Consider therefore the situation, where several of the groups wish to insert links into the technical specification. This can be accomplished in several ways using standard Web technologies. Either all links are placed in the same document, or identical copies of the specification are made for each group. Both solutions are unsatisfactory. In the former case, all groups will either have write-access to the specifications in order to add links, or possibly send their links to a single Web master. Both solutions have both overhead and the possibility of accidently corrupting either the specification or other groups' links. In the latter case, each group has their own copy. This too is unsatisfactory, if the specification at a later point changes, as n copies of the specification then must be updated. It could also be the case, that the specification is so important, that it may not be modified at all to prevent accidental changes.

These examples highlight that some things are not feasible or easy on the current Web. It is at this point quite unrealistic to replace the Web, and there is certainly no need, as the Web performs admirably for most purposes. However, it would be interesting to be able to perform the tasks described above better. One approach — the one taken by me through the course of my Ph.D. — is to try to improve upon the Web by adding an additional layer and leaving the layers below unchanged. This improved Web is what I term the augmented Web, partly in remembrance of NLS/Augment [39, 40] — a system that sought to augment the human intellect. I am not that ambitious, but I wish to augment the Web.

3.2 Lies Open Hypermedia Dead in the Wake of the Web?

It is the nature of my work that I place myself between two research communities, that of open hypermedia and that of the Web. Tim Berners-Lee never presented his World Wide Web formally to the hypertext community (at that time known as SIGLINK), though he held a demonstration of the system at the hypertext conference. Since then things have changed — the Web is big, and SIGLINK is today known as SIGWEB. So, has hypermedia research been rendered obsolete, maintained only by a few academic diehards? This question was discussed by Nürnberg and Ashman in [82] presenting the thesis that Web is the open hypermedia system of the future, and the antithesis that the Web is hardly a hypermedia system at all. These are extremist points of view, and

http://www.acm.org/sigweb/

the paper ends by reconciling thesis and antithesis into a synthesis. Briefly the paper concludes that there is room for both parties and that both can benefit from joining forces and learning from each other rather than focusing on differences.

Obviously, I think that there is room for both open hypermedia and the Web. Indeed, I do not foresee any immediate doom for open hypermedia caused by the Web. On the contrary I would argue that this is an *excellent* time to be doing research in open hypermedia, *precisely* because we now have the Web. As a topic for open hypermedia research, the Web has very nice properties. Most of the involved file formats and protocols are open (a relief for a community used to dealing with proprietary editors and formats), and there is a decent global naming scheme for documents and resources. Given the size of the content available (much of it fairly poorly structured), this is the time where the fruits of open hypermedia may be harvested. Hypermedia becomes first really interesting and useful with large collections [4, 5], because this is where good structuring tools really pay off. Not only is there a lot of stuff out there, there is also a lot of people using it (according to some of the latest estimates ~ 300 million, though this is notoriously hard to calculate). Organisations are actively moving their documents on internal and external Web sites. The Web has provided open hypermedia researchers with what we could scarcely dream of before — a unified fairly well-behaved document space with many users.

So what can we do with it? I along with many others have for the last few years been engaged in what I term Web augmentation. It is an approach to Open Hypermedia Systems development aimed at the Web, which seeks to combine the strengths of each technology to build a better Web.

3.3 The Vision of the Augmented Web

Web augmentation allows users to create and use external hypermedia structures imposed on Web pages (i.e. not previously present on the pages), that they themselves not necessarily have write-access to. This would empower users as they would be able to use the Web in a more flexible way. Given Web augmentation tools, it becomes possible to annotate, link, and otherwise structure all Web pages. Proper tools would also allow users to share their links with others, or professionals to publish their work to their colleagues, clients, or readers.

Revisiting the cases in Section 3.1, some things becomes possible or easier with Web augmentation tools. The following scenarios are all possible today.

> Web journalists can pinpoint their links to exactly what they want. They can add annotations to documents, highlighting inconsistencies or weaknesses in the text. They can create guided tours to help their readers get an overview of a topic. As time goes by, their hypermedia structures can be updated to reflect new material. With tools like this, there is suddenly room for trail blazers[1] on the Web, rather than collectors of links to other pages.
>
> Professionals relying on technical or legal documentation can now suddenly all share the same documents. The documents are not changed by the users, as they create the links and other structures they need in their work. Each collection of hypermedia structures can be viewed on its own, or combined with with other collections. They can be exported and mailed

[1] Trail blazers: In Bush's *As We May Think* [23], professional authors of hypermedia structures.

> to others. Using robust location specifiers [87] it becomes possible to have links meaningfully survive document revisions.

One concern that is often raised in this context is that of intellectual property (IP): is it OK to modify other people's Web pages? In the Coconut project, this question was put to the legal department of Tele-Danmark Internet. According to them, it is permissable under Danish law, as long it is clear that such modification is taking place. As Web augmentation generally requires users to install special software or to configure their computer in a special way, the users can be expected to be aware that modification may be taking place. IP laws vary from country to country, and this argument may not hold water everywhere. It should be noted though, that Web augmentation in technique if not in effect is a *presentation* — the original Web page is not modified, but the presentation of it on the user's machine is.

While not all problems of the Web can be addressed through Web hypermedia augmentation (e.g. "Error 404: Page Not Found"), the lessons and techniques learned in the open hypermedia community could very well benefit the users of the Web. Let us therefore consider, what Web hypermedia augmentation is.

3.4 Open Hypermedia Web Augmentation Tool: A Definition

> A tool is a open hypermedia Web augmentation tool, if it through integration with a Web browser, a HTTP proxy or a Web server adds content or controls not contained within the Web pages themselves to the effect of allowing structure to be added to the Web page directly or indirectly, or to navigate such structure. The purpose of such a tool is to help users organise, associate, or structure information found on the Web using hypermedia structuring mechanisms. This activity may be done by a single user or in collaboration with others.

Most Web augmentation tools operate solely on HTML, but as demonstrated by Webvise [56] and Mimicry [20] [P2], other media types can be hypermedia enabled, such as video (see Section 6.2 for further details on Mimicry). As the different media handlers grow more advanced and exhibit richer APIs (Application Programming Interface), it becomes possible to augment more of the Web.

Already, there exists not only research prototypes, that demonstrates the possibilities of Web augmentation, but also an increasing number of commercial products aimed at this area (some of which are described in more detail in Section 5.2). The field of Web augmentation is an intriguing one, that requires attention both to hypermedia research and the current state of the Web.

Chapter 4

Hypermedia and the Web

This chapter describes the foundation for the fields within which I have worked and motivates my work by examining hypermedia with special focus on open and collaborative hypermedia, as well as the Web. An understanding of what constitutes hypermedia and open hypermedia in particular is the necessary foundation of my work. Likewise, the history of the Web and its current state is needed to establish why augmenting it with open hypermedia might be a good and feasible project.

4.1 A Brief History of Hypermedia

Hypertext as a term was introduced in the sixties by Ted Nelson. The classical understanding of the word is a collection of documents (or "nodes") interconnected by links, so that a user easily can go from one place to another, as well as create links between documents. Today, hypertext (or hypermedia) is not just about links anymore, and it is perhaps better to redefine hypertext to reflect this. A modern but vague definition of hypertext could be the combination of content and structuring of that content.

Some argue, that the first example of hypertext is the Torah with its rich tapestry of interlinked annotations. However, the first description of a device that through electric or mechanical means provided what we today understand as hypermedia functionality was the hypothetical Memex, invented by Vannevar Bush in his landmark article "As We May Think" [23] published in the *Atlantic Monthly* in 1945. The Memex allowed the reader to speedily transverse a vast corpus of literature, to annotate pages, and to store the trails taken through this landscape of text. The Memex had near infinite storage capacity using microfilm, and new collections of texts and trails could easily be added or shared with other Memex owners. Proficient Memex users would become *trail blazers*, discovering new relations between texts and publishing them for other readers to use. This hypothetical machine formed one of the foundations for hypermedia research [68].

4.2 The Classical Monolithic Hypermedia Systems

The Memex was more than a glorified electric library, it was a machine designed to help people utilise associative thinking, to help people see new relationships where none had been apparent before. The idea to create a device to help people think better

is a radical one, and one that was taken up by two luminaries in the sixties. Douglas C. Engelbart and Theodor H. Nelson. Their work took two different directions: Engelbart as the innovator and creator, and Nelson as the visionary.

Engelbart's NLS/Augment [39, 40] was the first real hypermedia system. This ground breaking system provided its users with a collaborative, multi-window system (with video conferencing!), where everything could be linked to, and where the main interaction with the system was done through a mouse, another Engelbart innovation. The expressed purpose of NLS/Augment was to augment human intellectual capabilities.

The system designed by Nelson were no less impressive — he coined the term "hypertext" and envisioned the Docuverse [80], where *all* texts would be instantly available. This was to be realised through Project Xanadu — an extremely ambitious information infrastructure/hypermedia system that would take care of editing, versioning, backup, micro payment, typed hypermedia structuring, copyright, distribution, publishing, etc. — many issues still not solved satisfactorily today, nearly forty years later. The Xanadu system has yet to materialise though the implemented parts of it recently was released into open source. In some regards, the World Wide Web can be considered the actual Docuverse, though it is primitive compared to what Xanadu might have been.

The seventies and eighties saw the creation of several important hypermedia systems. KMS [1] became the first system to feature *large* hypertexts (used by the American navy). Intermedia [106], developed at Brown University for educational purposes had several interesting properties, and stands, at least in this author's opinion, as the pinnacle of monolithic hypermedia systems. Among its many advanced features were:

- Multi-user access to documents and hypermedia structures
- Access rights to read/write/annotate.
- Full fledged WYSIWYG applications
- Extensive undo/redo functionality
- Collections of documents ("corpus")
- Collections of (externally stored) hypermedia structures ("Webs")
- Span to span bidirectional links
- Graphical link browsers
- A large, object-oriented application framework to facilitate extensibility

A quite different but also impressive system was NoteCards [61] developed at Xerox Palo Alto Parc. Based on a metaphor of 3×5 cards, NoteCards was designed to be "a general purpose idea processing environment" [61, p. 45], intended for individual rather than collaborative use (see [94] for a discussion on the work of making NoteCards collaborative). One very nice feature of NoteCards was the Browser card, which was both a graphical representation of a link structure, an editor for said link structures, and an entity which could be linked to as any other card. NoteCard was developed in Xerox Interlisp, and could thus easily be modified and extended by the Lisp savvy. This allowed for great flexibility and supported experimentation with new forms of hypermedia structuring mechanisms, enabled NoteCards as a hypermedia testbed for further

experiments such as support for collaboration [94]. The lessons learned in the development of NoteCards was crystallised in Halasz' famous "seven issues" [60], where he identified areas that should be covered by new hypermedia systems.

4.3 Open Hypermedia

While individually impressive, systems such as Intermedia or NoteCards suffered from some problems. To be useful, they had to provide their users not only with hypermedia functionality (at which they excelled), but also with good editors (such as word processors, spreadsheets etc.). For specific purposes, such as an educational setting (Intermedia) or idea generation/structuring (NoteCards), they did very well, but still these hypermedia systems did not become widely used. One of the major obstacles to widespread acceptance was identified by Meyrowitz in [78] as the lack of support of the editors already in widespread use. Most organisations rely on standardised tools, such as word processors, databases, spreadsheets, CAD systems, etc., and unless the payoff is high, they are understandably hesitant to migrate to other tools. The monolithic hypermedia systems could only provide hypermedia functionality to their own editors and file formats, and thus required users to move exclusively to these editors. With the personal computer revolution came a lot of new applications, and the hypermedia developers could not keep up with the various new editors supported on the personal computers. The end result was that the hypermedia applications did not become as widespread as they might. Meyrowitz realised this, and suggested that the correct course was to embrace the third-party applications and to offer hypermedia functionality for them, ideally to the degree where hypermedia functionality was as ubiquitous as copy and paste. This became known as "open hypermedia", the augmentation of third-party applications with hypermedia.

An added advantage of integrating third-party applications is that it allows people to create relationships between documents handled by different programs, that would otherwise be non-integrated. Thus, dissimilar applications are "knitted together" by the open hypermedia system.

Some of the current open hypermedia systems are (listed alphabetically)

- Chimera [7] (University of Colorado, Boulder, USA)
- Construct [52, 53] (University of Aarhus and Aalborg University Esbjerg, both Denmark)
- DHM [50], now Webvise [56] (University of Aarhus and Mjølner Informatics, Denmark) `http://www.mjolner.dk/`
- HOSS [81] (Texas A&M University, USA)
- HyperDisco [103] (Aalborg University, Denmark)
- Microcosm [63] (Southampton University and Multicosm Ltd., UK) `http://www.multicosm.com/`

All of which are represented in the Open Hypermedia Systems Working Group (OHSWG). See Section 4.4.2 for more details on this initiative.

Open hypermedia systems (OHSs) have in general many traits in common, which given their similar problem space is to be expected. Given the need to integrate third-party applications with different (often closed) document formats, it is in general impossible to store links etc. inside documents, and so OHSs utilise externally stored hypermedia structures. This necessitates some sort of storage of these structures, usually

handled by a hypermedia server. Likewise, a client is needed to retrieve the hypermedia structures, and display them somehow in conjunction with the document they refer to. Areas where the OHSs often differ are aspects such as approaches to distribution, collaboration, link markup, and extensibility. The sections below describes a few representative OHSs based on the findings of the first open hypermedia system.

4.3.1 Sun's Link Service

The first open hypermedia system was Sun's Link Service by Pearl [86]. The Link Service consisted of a link server, and an API (Application Programming Interface) for developers, so that they with small modifications of their applications could provide users with bidirectional links. The Link Service was an integrated part of the operation system (SunOS) and the windowing system (OpenLook), and was as such available to all applications running on the platform. The system provided users with shared or private collections of links, and support automatic or explicit link garbage collection.

Sun's Link Service failed to penetrate the marked, perhaps because the success of the system relied on third-party developers to actively modify their own applications (the reliance on OpenLook quite probably did not help, either), but in the above mentioned paper, Pearl identified some interesting issues with the development of open hypermedia systems:

User Interface Per definition an open hypermedia system relies on third-party editors to handle documents. There are two user interface aspects in this: firstly, there must be an interface to interact with the linking service; secondly, there must be some "highlighting" identifying the linked objects.

Document control Another common theme in open hypermedia is the ultimate lack of control of documents and their whereabouts. Unless documents reside in a document management system, they can be modified, deleted, or moved without the hypermedia system's notice.

Link consistency Related to document control, links may become inconsistent if one of the constituent documents disappear, is moved or modified, or is otherwise unavailable.

Locating endpoints Apart from the problem of disappearing or changing documents, it can in itself be a problem to locate an endpoint in a document type not designed for such uses. Especially if (or rather when) the document is modified.

Distribution Traditional monolithic hypermedia systems typically functioned within a context of *one* file system. An open hypermedia system may well contain documents residing on other, possibly remote, file systems. This raises questions with regards to document control and link consistency — should the link be marked as invalid (or even deleted) if the connection to the server holding a constituent document is down?

Collaboration It is natural to collaborate on the material held within a hypermedia system, and this should of course be supported by the system. This raises questions about sharing, permissions, locking, etc.

Versioning Both documents and hypermedia structures can be versioned. While hypermedia versioning presumably would be under the hypermedia system's control, document versioning can be a problem. E.g. what should a link point to?

The version of the document it originally pointed to, or the latest version? What if the latest version does not contain the link's endpoint?

It should be noted, how these issues relate to respectively the client and the hypermedia server. User interface and the location of endpoints are clearly client side problems. Another more interesting case is link consistency. Most implementations of open hypermedia systems would have the client determine link deterioration, as this is most often discovered upon link decoration. As long as the server is able to store location specifications of a sufficient general nature, the client should be able to attempt to regenerate the link by locating the lost anchor. Of course, if the location specification of the hypermedia model employed is very specific, such as byte offsets, this is not possible. Link consistency problems related to lack of control of documents is however a general systems characteristics.

I will revisit these issues, as I go through some of the open hypermedia systems below, as they are still valid, and illustrate some different design decisions in these hypermedia systems.

4.3.2 Microcosm

The Microcosm Link Service [33, 42] from Southampton University was the first open hypermedia system, that is still in development and use. It is an innovative and (in some ways) unconventional hypermedia system. The hypermedia group at Southampton (and later the company Multicosm) has excelled in the integration of many third-party applications, a topic described in more detail in Section 4.3.8.

One of the innovations of this system was *generic* links — links with a destination endpoint with a specific location, but with a source specifying only the content of a source endpoint, rather than its location. One of the problems with much hypermedia linking is that link authors have to author *all* the links, so that if new documents are added to the corpus, they are initially "unlinked". By creating links that rely on e.g. keywords in text, it becomes possible to reuse existing links in new documents. A trivial example of the usability of generic links would be an electronic dictionary, allowing the user to rapidly access definitions of words and terms.

The Microcosm approach called for an aggressive inclusion of third-party editors ("viewers"), many of which could not be made Microcosm "aware". In order to support these kinds of applications, some functionality such as automated highlighting of anchors were eschewed. Anchors in a document could instead be found on demand or displayed in a separate window.

A particular feature of the Microcosm architecture is the filter model. Messages from e.g. clients are processed by filters that can stop, process or in other ways modify the message by e.g. adding the anchors present in a given document. These filters can be added or rearranged by the user.

Returning to the issues raised by Pearl in Section 4.3.1, the design of Microcosm has some interesting consequences. Interface-wise, the reliance on generic links means that there often is no markup of links. Indeed, the creators cite this as a strength of the system. On the other hand, it makes it much easier to integrate editors, and thus vastly expands the number of applications that hypermedia functionality can be added to. Microcosm features a little hypermedia button added to all application, from where the user can follow links etc. This is for instance possible in Windows, and gives the user immediate access to hypermedia functionality. Document control can handled by a document manager, which addresses some of the concerns with regards to document

control. However, if the documents can be retrieved outside of the system, the possibility of corrupted documents (from the view point of the hypermedia system) remains. This is however less of a problem in the context of Microcosm (baring the removal of documents), given the focus on generic links. Link consistency is closely related to this. While a generic link will work in all documents, as long as there is a selection matching the link, the destination document (or the specific location within that document) may well be gone. Given the Microcosm approach, a problem such as locating an endpoint is circumvented, as many endpoints (especially to unaware editors) are whole-component. With specific anchors, the default Microcosm approach is to use byte offsets [32, p. 212] to locate anchors. Such an approach is quite fragile, and can be supplemented by "macro" (for applications, that support such) to locate the selection. With the work on Microcosm TNG (The Next Generation) [46], Microcosm can handle distributed document, link, and service location. TNG also incorporates a notion of sessions, wherein users can specify, share and collaborate on selected services and applications, that is nodes and link collections. Versioning in Microcosm is described in [76], as layers of applications. This arrangement allows user to freely access nodes and links from different versions, but does not support merging.

4.3.3 Devise Hypermedia

The Dexter Hypermedia Reference model (described in further details in Section 4.4.1) inspired the development of the Devise Hypermedia (DHM) at the University of Aarhus. DHM was originally developed for the Macintosh and Unix platforms using the object-oriented language, BETA. It saw much development during the EuroCODE project, where it was used in a large engineering setting [47, 49]. It has been used extensively as a research prototype, and has in its development history been a cross-platform hypermedia system, a collaborative hypermedia system [49, 50], a tailorable system with an embedded interpreter [51], hypermedia basis for an 3D collaboration system [22, 79] and in its current incarnation the Windows based Webvise hypermedia system integrating Microsoft Office and Internet Explorer [56].

While based on Dexter, the work involved with the implementation of DHM necessitated clarifications, extensions, and departures from the original model: The Dexter model adamantly forbids dangling links, which DHM permits; composites contain in Dexter their constituent components, where they in DHM may contain references instead; anchors to whole components are not specificed in Dexter, which they are in DHM. By design Dexter left the within-component layer largely unspecified save the anchoring interface as to concentrate on the core hypermedia functionality of the model, but for an open hypermedia system the within-component or application layer is quite important. Dexter assumes that components are controlled by the hypermedia system, but in open hypermedia documents etc. are usually owned by various other applications, and this had to be encompassed in DHM. It should be noted that Dexter in other aspects, such as anchoring and presentation specifications, was found to be well suited for open hypermedia. One of the main focuses of the development of DHM has been client integrations, as DHM has been used in various research projects where certain applications has been used by the participants. This has resulted in integrations with Microsoft Office applications, as well as technical software, such as CAD systems. The Webvise client was, to the author's knowledge, pioneering in utilising the Microsoft Internet Explorer's COM (Component Object Model) interface to integrate the Web, an approach later also used by the Arakne Environment.

DHM is a conceptual three-layer hypermedia system. The storage layer has varied

through the history of the system, originally a object persistence store was used — the current version utilises file based storage. At the structure level the system currently supports *n*-ary, bidirectional links and guided tours. Client wise the system has it own dedicated client, the Webvise Client, but has also supported the DHM/WWW [48] and Navette [16] clients, as well the first version of the Arakne Environment [17] [P1].

As for Pearl's issues, the DHM systems have always stressed the user interface, which has resulted in some highly integrated interfaces in especially Microsoft applications, where the hypermedia functionality is made available through menus, and where anchors are highlighted in the various documents. Documents are outside of the system's control, which leaves the possibility of link inconsistency, but depending on the integrated application, Webvise has fairly robust anchor specifiers. While DHM has at times been a collaborative, LAN (local area network) distributed hypermedia system [50], the current incarnation, Webvise, is not aimed at collaboration, save by exchanging interchange files. The exchange of such files is quite well supported, as they by the client can be stored on WebDAV compliant servers (see Section 4.7.4 for more information on WebDAV, and Section 7.3 for a discussion on how this may help scalability), and their retrieval in a Web browser will on a computer with the Webvise client installed automatically launch the client, and display the hypertext. DHM has no provisions for versioning, though it can be used in conjunction with a system supporting document versioning, as done by the High Road Demonstrator in the EuroCODE project.

4.3.4 HyperDisco

HyperDisco comes from a family of hypermedia systems developed at Aalborg University: Hyperbase 0–2 [100], Hyperform [101], the Emacs HyperText System (EHTS) [99], and HyperDisco [102, 103].

A characteristic of these systems is the architectural focus on distribution, collaboration, and hypermedia database systems. Few systems, save perhaps the hypermedia systems developed at Texas A&M University (see Section 4.3.5 for a description of HOSS), has focused as much on hypermedia infrastructure. Indeed, HyperDisco may be considered a "meta hypermedia" system, to wit:

> The HyperDisco project is about design, development, deployment and assessment of OHSs. The overall goals of the project are to work towards innovative new OHSs and to try to influence the development of future generations of OHSs for the Internet. [104, p. 296]

The HyperDisco hypermedia system has several interesting properties. It is essential a three-layered architecture with tools (integrated or native), tool integrators (usually one per user), and workspaces. A workspace embodies hyperbase and collaborative functionality, and can be distributed over the Internet, and users can choose which to operate on, assuming they have the proper permissions. Returning to Pearl's issues, HyperDisco addresses many of these concerns. HyperDisco integrated ten applications by October 1999, most well-known of which is XEmacs, which is integrated with menus and markup. HyperDisco can handle documents stored outside of the workspace, as well as within it, and certainly in the latter case can exert document control. This also to a large degree answers the concern of link consistency. Certainly if the document is held in HyperDisco's control, link consistency should be assured. As HyperDisco anchors are extensible, the ability to regenerate broken anchors lies with

http://www.xemacs.org/

the various client. HyperDisco handles versioning at the data model level by providing a Version Control class, that (through multiple inheritance) can be subclassed by any component. This is used mainly for versioning of nodes, while other components (such as links, composites, etc.) are not (at least at time of writing [102]) versioned. Distribution and collaboration are areas, where HyperDisco shines. As described above workspaces can be distributed Internet-wide, and users of the system can freely collaborate within these workspaces.

HyperDisco along with other systems (especially HOSS described in the following section) informed the creation of the Construct hypermedia architecture, described below (Section 4.3.7).

4.3.5 HOSS

HOSS [81, 83, 84] grew out of a series of hyperbases and hypermedia systems developed at Texas A&M University: PROXHY [67], HB1/SP1 [90], HB2/SP2 [91], and HB3/SP3 [71]. An interesting feature of HOSS is the focus on providing hypermedia functionality on the operating system level. Usually hypermedia systems are considered to belong to the application level, giving the user tools to interact with and structure information. Another innovation of HOSS is the separation between data, structure, and behaviour. Separating data and structure is arguably what hypermedia is all about, and this system takes it one step further by allowing behaviour to be defined separately. In contrast, most hypermedia systems have more or less fixed e.g. link transversal behaviour. By opening up behaviour, and relegating both structure and behaviour to the operating system level (where data has been all along), this approach opens up some radical possibilities. A hypermedia operating system would be very flexible, and would provide users with superior performance when working with structures, as the system would be able to prefetch needed data or structure, cache behaviours and structure, automatically handle structure etc.

http://www.csdl.tamu.edu/

A prototype of this approach has been implemented in the HOSS system. HOSS runs on SunOS, and provides services for handling data, structure, and behaviour. The system offers an open set of behaviours (“Sprocs”), in the version described in [84] navigational and taxonomic hypermedia, as well as annotations. Applicationwise, the system integrates a spatial text editor [31], an animation tool, and a reviewing tool. In addition, HOSS also provides launch-only integration for non-integrated tools. The integrated tools use the object store provided by HOSS for storage. As the HOSS storage layer offers versioning, this is also available for structures and the data stored within the storage service. A fairly unique (certainly the author is aware no other system providing this) client is `Hsh`, the HOSS shell similar to Unix shells, but aimed at controlling the hypermedia system. HOSS is a hypermedia-in-the-large system, and can thus handle distribution. From [83, 84], it is unclear how collaboration is supported. With regards to link consistency, by virtue of storing at least some documents within its own object store, the system should be able to provide some consistency guarantees, though this is largely conjecture on the part of the author.

http://www.sun.com/

Apart from the idea of integrating hypermedia on the operating system level, a very interesting feature of HOSS is its development tool. HOSS’s designers stress the importance of an open set of services and to that end provide the developer with the PDC. The Protocol Definition Compiler generates, based on an IDL like specification, C libraries to handle IPC (interprocess communication) between HOSS components. The speeds up development time, as programmers are freed from the tedious and error prone work of developing such code. The work on hypermedia development tools has

been continued at Aalborg University Esbjerg in the form of CSC [105], a development tool for new Construct (Section 4.3.7) hypermedia services.

4.3.6 Chimera

Chimera [7, 8] is a hypermedia system developed at University of California, Irvine, and University of Colorado, Boulder. It has seen industrial use [5], and has as such been the topic for scalability studies [3, 4, 5], as well as some initial studies on XLink [64]. From its beginning, the main focus of the Chimera system has been large software development projects, which has, among other things, informed the work on scalability. The Chimera architecture is three-layered with a relational database serving as storage. An interesting feature of Chimera is the ability to serve Java applets to users, so that they can access and manage the system without installing special software locally. Hypermedia wise, a unique feature of Chimera is the concept of "view". Open hypermedia systems in general handle viewers/editors as programs able to display a document, and are often able to launch the viewer associated with a media type. Anchors are in such a model associated with the document, and are presented depending on the abilities of the viewer. Chimera has a different approach to this, as anchors are associated to views, that is the combination of a viewer and a document. The implication is that Chimera can present the same document differently (with different anchors), depending on the viewer used. One example of this would be a spreadsheet presented either as a matrix or as a graph.

Returning to Pearl, the interface for managing hypermedia structures lies, depending on the level of integration, in the Chimera Client or in the viewer. Chimera exerts no document control, and can therefore as most other hypermedia systems get in trouble with document and link consistency. As anchors are view specific, their appearance and location depends naturally on the abilities of the viewer. Chimera supports collaboration through and distribution of HyperWeb servers through HyperWeb managers, which (among other things) handles the location of HyperWeb servers. Versioning, while outlined in [97], has not been implemented.

4.3.7 Construct

Construct is one of the results of the Coconut project. It is the codebase descendant of HOSS [81, 83, 84], HyperDisco [102], and DHM [50], designed and developed by the people (among others!) behind these systems. It is compliant with existing OHSWG standards, along with several extensions, such as compositional and spatial hypermedia. It is also the backend of the Arakne Environment.

Design wise, it has inherited much from its ancestors. It has a clear three-layer design with an open and extensible set of services. The architecture is highly symmetric with regards to the three layers, which is illustrated by the overall process architecture seen in Figure 4.1. This process architecture is found at all layers — clients communicate with a structure server through a virtual server, servers communicate with clients through virtual clients and so on. The core holds "actual" functionality — thus a navigational hypermedia client uses the Nav core to operate on navigational components. The architecture with virtual servers and clients provides multiplexing and leaves room for other transport layers — the current Construct uses XML over TCP/IP sockets, but this may well change in the future, as OHSWG moves forward, as outlined in Section 6.3.

Hypermedia structure wise, the Construct architecture currently supports navigational, compositional, and spatial hypermedia. Due to its extensible nature, this is however not a closed set. With the advent of the CSC tools [105] from Aalborg University Esbjerg, the development cost of new hypermedia services will be greatly diminished. As an example of the data model of Construct (and in this instance of OHSWG), see Figure 4.3. The navigational data model is discussed in more details in Section 4.4.2.

Construct supports collaboration at the server level, mainly through the notion of "sessions". What this collaboration support means at the client is described in Section 6.4. To address link consistency, the OHSWG navigational hypermedia model has a flexible and extensible set of location specifiers. How this is utilised, depends on the client. The current version of Construct does not support versioning.

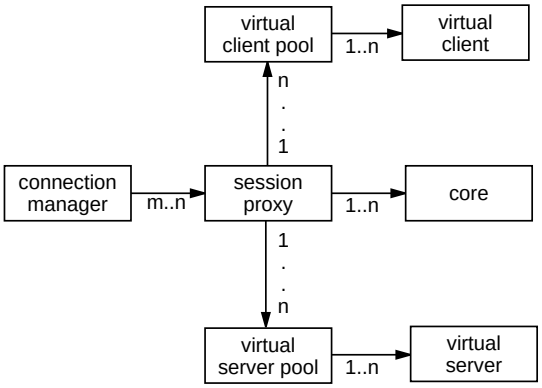


Figure 4.1: The Construct Process Architecture (from [52])

4.3.8 Integrating Third-party Applications

The defining characteristic of open hypermedia is the integration of third-party applications. It is also one of its greatest problems, as it is often hard to accomplish satisfactory integrations. The first to write about this was Davis *et al* in [34], and it has later been covered by Whitehead in [95]. Application integrations differ in the degree with which the hypermedia structures can be made visible and manipulated directly in the application user interface. The most basic integration is "launch-only", that is a situation where the hypermedia system launches an application to display a specific document, but has no further control over the application. In such a situation, only whole component (i.e. the entire document) references are possible. Fortunately, many applications support various levels of scripting and tailoring, and by such means it is possible to create quite sophisticated integrations, where fine-grained addressing of the application's document becomes possible, and where much of the hypermedia interface may reside in the application. Applications with sophisticated integrations include (X)Emacs, Microsoft Office, AutoCAD, MicroStation, Framemaker, and various Web browsers.

In the course of the author's work, the applications to integrate have been Web browsers, and the experiences in this context can illustrate the various means, with which such an integration may be accomplished. This is described in the following

section, and considerably more detailed in Section 7.1.

4.3.9 Open Hypermedia on the Web

The open hypermedia community is in some ways uniquely well suited to take on the Web with hypermedia. The community has extensive experience of supplying other applications with hypermedia functionality, and the Web has qualities (outlined in Section 3.2) making it an "easy target" for integration.

This section focuses on the various approaches taken to support hypermedia integration with the Web, and will so in passing describe some of the systems that employ such approaches. It should be noted that there are also open hypermedia systems, that utilise the Web without actually performing Web augmentation. Chimera [7] actively uses Web browsers, and Web-aware HyperWeb servers to provide users with hypermedia clients and maintenence tools. These tools are however used for controlling the Chimera system, not for hypermedia augmentating Web pages, and in this sense the Chimera system uses the Web more as a transport layer.

Given the general architecture of the Web (see Figure 4.5), it is fairly obvious that there are basically three "lines of attack", when wanting to gain access to the Web pages displayed in a Web browser. It can be done at either the Web server where the pages reside, at an intermediary Web proxy, or in the Web browser. The possibilities and consequences are outlined below.

Server side

It is quite easy to add content to a Web page through a Web server. This content can be dynamic (relative to more static Web pages) and easily be switched on and off by the user through the use of cookies etc. Often this kind of augmentation is achieved through CGI (Common Gateway Interface) scripts, with Walden's Paths [45], one of the classic guided tour tools for the Web, as a prototypical example. The guided tour navigation menu is dynamically added to the pages that constitute the contents of a path. There are however some drawbacks to this approach. If one wishes to augment Web pages from more than one Web server, these "foreign" pages must be modified in some way. One typical approach is to modify all links encountered to invocations of CGI scripts with the original link as an argument. However, the use of e.g. frames or bookmarks in the Web browser defeats the modification of links, and thus all actions by the user cannot be handled through this approach.

Similarly, the authoring of hypermedia structures becomes difficult, as there is no interface, save the Web page. The common approach to this is either to have a separate authoring tool (this is the case of Walden's Paths), or to provide special Web pages with forms allowing the authoring of e.g. links, though the latter approach quickly becomes cumbersome.

The two greatest advantages of the server side solution are that the user does not have to modify browser configuration other than using a specific Web site, and that the services can be accessed from any Web browser.

Proxy side

To maintain the Web augmentation system's access to the Web pages displayed in the user's Web browser, the most obvious approach is to use a proxy. Proxies are widely

used for caching purposes[1], but can also be used to modify Web pages. The prototypical example of this is the Microcosm Distributed Link Service (DLS) system [25]. In these cases the proxy acts as a hypermedia client, retrieving links etc. from link bases and inserting them into the Web pages. The great advantage is that the system is, once the Web browser has been configured properly, always available. This is ironically also one of the greatest drawbacks — it is quite plausible that the user does not wish to constantly use the proxy, as the querying of link bases inevitable will slow down the process of browsing. Indeed, this drawback (among others) later led the Microcosm group to create the AgentDLS system [24], where links were displayed alongside the primary Web page in a periodically refreshed Web page rather than being inlined, boosting performance considerably. This approach raises the question of whether this kind of interface results in too many focus shifts, as users try to correlate a Web page with the "links page".

Another hypermedia proxy is the DHMProxy [56] created by the Coconut project, using the DHM and Construct servers. An interesting feature of the DHMProxy was the method of link decoration (i.e. the act of inserting links etc. into the Web page), as it was, for Web browsers that supported it, like that of Webvise or the current Arakne Environment. Based on the information retrieved from the hypermedia server, the proxy inserted into the Web page Javascript code, which upon arrival was executed and performed the link decoration. This was done to circumvent the problem described below with dynamic Web pages.

The reservations with regards to hypermedia authoring noted above also hold for proxy solutions, e.g. DLS used a separate Web page for link authoring. Of course, one can opt to have a client running on the user's computer. The early versions of the Arakne Environment provided authoring, but employed the DHMProxy for link decoration. In order to use a proxy solution, the user has to setup the Web browser to use that particular proxy. This has some problems: The user may not want (or may not be able) to make this configuration change; the use of a special proxy may collide with the use of another proxy[2]; and the user will feel the performance hit of the hypermedia proxy, whether the current Web page has any extra hypermedia content or not.

Client side

It is also possible to modify Web contents at the client. The first system to do so was the DHM/WWW system [48], and the client-side approach has also been taken by its descendants, Navette [16], and finally the Arakne Environment [17] [P1], as well as Webvise [56].

Historically, there have been two approaches: Pre- and post-render page modification, that is before and after the Web browser has displayed the page. DHM/WWW, Navette, and the earliest incarnation of Arakne all utilised a pre-render approach, DHM/WWW by retrieving the page, modifying it, and generating Javascript to output it to the Web browser; and Navette by acting as a local proxy. Both instances can thus be argued to be quasi-proxy solutions, both with the great disadvantage of having to compute links, etc. before rendering the Web page. In contrast, both Webvise and the current Arakne Environment use post-rendering page modification. There are several advantages to this approach. The first advantage is that of performance, as pages are displayed in the Web browser as fast as they would without the hypermedia application. The second advantage is the ability to more robustly handle "difficult" HTML. Many

[1] Indeed, ISPs often use proxies transparent to their users.

[2] The Coconut proxy addressed this by allowing users to specify a proxy for the Coconut proxy to use.

Web pages are today highly dynamic in the sense, that they contain a lot of Javascript which upon arrival is executed and modifies the page. This is very difficult to catch efficiently pre-render, but easy after the Web browser has rendered the page.

The great advantage of a client side solution is that authoring becomes very easy to support. Client side necessitates a hypermedia application running along the Web browser[3], and this application can of course also accommodate authoring. On the flip side, it forces the user to shift focus between two application during authoring as well as hypermedia applications other than simple link traversal.

The great disadvantage of client side integration is that, given the current generation of Web browsers, the choice of Web browser (and hence operating system and computing platform) becomes quite limited. Currently the post-render approach can only be supported by the Microsoft Internet Explorer under Windows. This is highly unfortunate as it locks many users out. Until other suitable Web browsers (such as Mozilla) appear, the situation is unlikely to change. See Section 7.1 for further discussion on the topic of Web browsers.

4.4 Hypermedia Standardisation Efforts

Throughout the history of hypermedia, there have been several attempts to standardise hypermedia, both on a conceptual level and on an interchange and interoperability level. This is important work for several reasons. Firstly, by clearly agreeing on what hypermedia is, it becomes easier to focus and compare research efforts. Secondly, the various hypermedia systems gain, if it is possible to either interchange documents, or perhaps even get different hypermedia systems to work together. Thirdly, if there exists a standard supported by several hypermedia systems, this standard becomes attractive for other developers to support, thus spreading general hypermedia support to more applications, and thus reinforcing development, as described by Whitehead [96].

Standardisation efforts are, however, not without pitfalls. Make a standard too big and complicated, and it is unlikely to catch on, as the investment to become standard compliant is too big. An example of such a standard is SGML, which while certainly used in large organisations, was not seeing as widespread use as it might. This was largely due to the very high development cost of producing tools compliant with the very comprehensive standard. Few tools were developed, and these tools were quite expensive. With the introduction of XML, a simpler version of SGML, the cost of adopting was lessened dramatically to the extend, where parsers and other tools can be obtained freely and open source. Today, XML is one of the new powerful paradigms of Internet distributed computing, and is being used extensively in many areas.

In the context of open hypermedia, a standard should either be simple conceptually, or modular, so that the hopeful standard adopters can incrementally implement what they need. As there in open hypermedia are several well established systems, a standard that closes one out by specifications that cannot be met by this system is likely to be met with resistance. Again, a modular standard is preferable, so that areas of contention — such as composites or collaborative support — can be adopted by those systems that can support them, while still allowing other systems to use other less contentious modules of the standard. Furthermore, this would allow others to “pick and

[3] Systems may integrate directly into the Web browser, as the Internet Explorer add-ons (see Section 5.2), but they are still separate processes. The Webvise client has experimentally been integrated with the Internet Explorer in this fashion.

choose" from the standard, solving their problem space without having to implement unnecessary or unwanted functionality.

4.4.1 The Dexter Hypertext Reference Model

The Dexter Hypertext Reference model [62] introduced a common framework for hypermedia systems. Its contributions were manifold. Among the more crucial were a general hypermedia architecture, the generalisation of the link from binary to arbitrary arity (i.e. n-ary or multiheaded links), the realisation that nodes, links, and composites are separate first-class components, the generalisation of the composite, and the clear specification of what the responsibilities of a hypermedia system is.

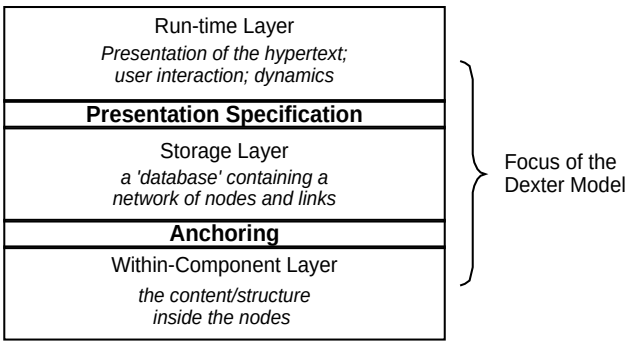


Figure 4.2: The General Dexter Architecture (from [62])

The Dexter architecture is summarised in Figure 4.2. The focus of the Dexter model was mainly on the storage layer and the entities stored within it. The objects stored in the storage layer are summarised in Table 4.1. The hypermedia model of Dexter is quite simple and elegant, and a novel aspect of it is that links and composites just as atomic components can be the target for links. The composite is used to contain other components, including other composites (though not itself).

The Dexter model does not concern itself with the applications wherein the atomic components should be displayed. This is all a part of the within-component layer. The focus is on the storage layer and the interplay between the run-time layer, containing a running hypermedia system used by a user, and the storage layer, containing the databases serving components to the hypermedia system.

Much is left out of the Dexter model — there is no collaboration; editors are opaque; as are presentation specifications and anchor values. This is however one of the model's strengths: It does not get bogged down in implementation specifics. Thus, it was able to inspire and provoke a lot of thinking [71] and (especially at Aarhus) development of new hypermedia systems. See Section 4.3.3 for a more detailed discussion of the Dexter based hypermedia systems developed at Aarhus.

One of the goals of the Dexter model was to facilitate interchange between hypermedia systems, and some work has been published on that topic of KMS to Intermedia [69]. The problem of interchanging files between too different hypermedia systems is that there invariably will be "hacks" in order to accommodate hypermedia structures not supported by both systems — a simple example is the translation of a one-to-many

Components	All components have a globally unique identifier (GUID), a list of anchor indexing into the component, a presentation specification, and a list of attribute/value pairs.
Atom	The "nodes" of the Dexter model. Known in other hypermedia systems as cards, frames, documents, etc. A Web page is a node. Opaque in the Dexter model.
Link	A link contains a list of specifiers
Composite	A composite contains components, including other composites.
Entities stored in component info	
Specifier	A specifier contains an identity unique within the link, a specification of a component (often the GUID), an anchor id, a direction, and a presentation specification.
Anchor	An anchor contains an identity unique within the component, and an anchor value. The anchor value (opaque to the Dexter model) identifies a part of a component.
Presentation specification	The presentation specification determines how a component should be presented to the user. This is opaque to the Dexter model.

Table 4.1: Components and entities in the Dexter model

link into a system supporting only binary links. With such intermediary translations, it becomes difficult to have "robust" interchange, that is, a situation where a hypertext remains the same after being translated to another system and back. If this is not possible, the usefulness of interchange files are diminished, as they then serve more as an migration path than as a possibility of working together across different hypermedia systems. Section 7.3 deals with other uses for interchange formats.

4.4.2 Open Hypermedia Systems Working Group

The Open Hypermedia Systems Working Group [35] (OHSWG) was founded in 1994 by the different parties involved in the development of open hypermedia systems. Among the original goals of the OHSWG was to collaborate on third-party application integrations, as this has always been the great challenge to open hypermedia development and an area, where code sharing could be mutually beneficiary. Apart from this utilitarian goal, OHSWG also wished to identify what constituted open hypermedia and to devise a common standard, so that open hypermedia systems could interoperate. http://www.ohswg.org/

The work of OHSWG has over the years evolved into the Open Hypermedia Protocol (OHP), a data model and protocol for collaborative, open hypermedia. Today, OHP encompasses a growing number of protocols that covers hypermedia, such as navigational, compositional, and spatial, collaboration support, subscriptions, naming and location of hypermedia services, caching, and document retrieval. The scope of this text does not permit a comprehensive study of the entire standard, but the navigational data model can be seen in Figure 4.3.

Some aspects of the OHP navigational model are interesting in the sense that they

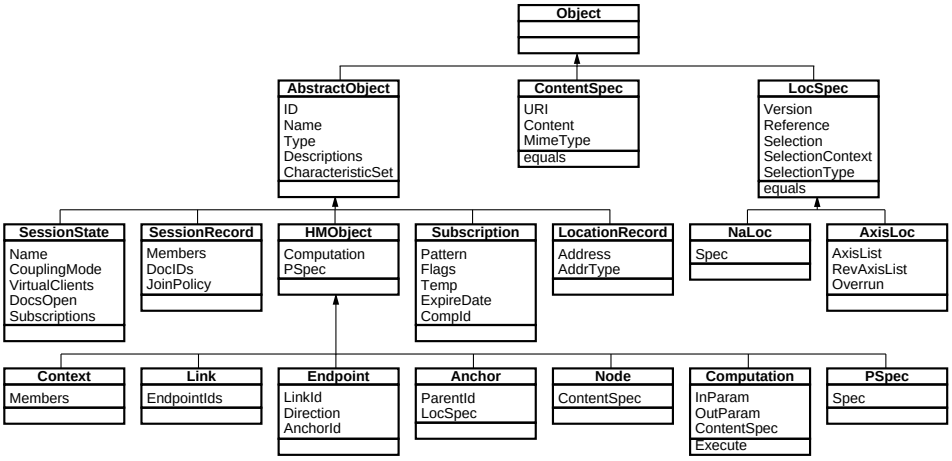


Figure 4.3: The OHSWG Navigational Data Model

are very general and allow for many kinds of extensions and innovative future uses. For instance all HMObjects has presentation specifiers, as well as computations and arbitrary key/value pairs ("characteristics"). A Context is the container for other HMObjects' IDs — including other Contexts. The parentId of the Anchor is the identity of the object containing it — this is usually a Node, but could just as well be *any* other HMObject (as HMObjects all have IDs). Thus, a link can point to any other HMObjects.

Aside from diverse hypermedia structuring mechanisms, OHP also supports collaboration through the notion of sessions. A group of people working together on hypermedia structures on a set of documents using various hypermedia tools embodies the session. A session is defined by a set of users, documents, and tools, and has a coupling mode (ranging from uncoupled to tightly coupled) and a joining police (e.g. a public or private session) associated with it. Through the session, each collaborator is visible to the other collaborators, as are his or her actions, depending on the coupling mode. If a collaborator so desires, special subscriptions can be set, so that specified actions taken by certain users result in notifications. The session is persistent, so that a group at a later point can resume work where it was left.

Collaboration in hypermedia was not introduced by the OHSWG. It is a fairly old field within hypermedia research, and is described in the following section.

4.5 CSCW and Collaborative Hypermedia

The fields of computer supported collaborative work (CSCW) and hypermedia share roots, as some of the first papers published on collaboration using computers were pioneer hypermedia papers "As We May Think" by Bush [23] and "A Conceptual Framework for the Augmentation of Man's Intellect" by Engelbart [39], published in respectively 1945 and 1963. Bush and Engelbart both wished to improve work — Bush in devising better ways to cope with a rapidly growing scientific literature, and Engelbart in helping knowledge workers to handle difficult problems better.

Work is rarely done by exclusively the individual; there is usually a social context within an organisation, that determines why work is done, what work is done and how it is carried out. This social context demands cooperation between workers, be it on the same level of the organisation or between levels. The purpose of computer supported cooperative work (CSCW) is to investigate and understand this cooperation, and to support it with computers. Bannon and Schmidt have written one of the better definitions of CSCW:

> an endeavour to understand the nature and requirements of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements. [11][p. 11]

The CSCW field is characterised by its diversity, and it attracts people from many disciplines. As noted in [10], the term CSCW itself is contested. What *is* cooperative work? Schmidt and Bannon use the social science tradition (reaching back beyond Marx and onwards) to define cooperative work as the activity people engage in when they work together and are mutually dependent in the sense that one cannot finish the work without the contribution of the other. This interdependence requires planning, coordination, allocation, scheduling, commitment etc. from the people engaged in the work. These activities necessary for cooperative work are under one referred to as *articulation* work. The different settings and conditions under which cooperation is found may vary tremendously, but articulation is always needed to make the cooperation work. By focusing on articulation, Bannon and Schmidt are able to come up with a general understanding of what cooperative work is, and how it may be supported by computers. They avoid a common pitfall of CSCW: focussing on the group. A good example of the predominant group focus is the term "groupware". Cooperative work is not limited to the group: people work together across boundaries — an example is networking between within and without organisations.

CSCW is a large field and the scope of this text does not permit a general overview. One area especially pertinent to my work (papers [15] [P4] (Section 6.4) and [6] [P5] (Section 6.5)) is the study and support of shared awareness. As described by Heath and Luff [65] in their analysis of work done by traffic controllers in the London Underground, maintaining shared awareness between co-workers can be crucial. Many systems has endeavoured to support this. The GroupDesk system [44], the NESSIE system [88], the Elvin system [41], the AREA system [43], and the BABBLE system [21] to mention a few. These systems try to support shared awareness through means such as messaging, chat, and event notification. Some of the awareness support require explicit actions by the users, such as sending a message to another user, others are implicit. The latter form is for instance the action of updating a file in a CVS repository, which gets broadcast to all users of the Elvin system, or joining a topic in BABBLE, which shows others users, that you are active. Some systems, notably AREA and Elvin, attempt to integrate with other applications, so that actions in these applications are broadcast to other users.

These tools are not only found within academia: a commercial and very widely used system (reportedly 3 million users) is ICQ, a "buddy list" program, where users may see when their friends are online and engage in asynchronous message exchange and synchronous chat.

`http://www.icq.com/`

In the context of shared or peripheral awareness tools, the topic of privacy becomes important. Not all actions taken by a user belongs in the public, and the user will (often rightly) fear a system facilitating "Big Brother". When people work together,

they normally share an understanding of what is acceptable to the participants and what is not. If not the boundaries between people are respected in the system, trust is eliminated and that can prove disastrous to continued cooperation. Clement gives in [29] a great deal of thought to the privacy issue in the context of media spaces (such as Portholes [38]), but his conclusions and warnings are applicable to the entire CSCW field. To return to Bannon and Schmidt:

> *visibility must be bounded* [...] Deprive workers of that capability, and they will exercise it covertly [11][p. 35]

Hypermedia is per definition used to structure information. This can of course be very usable for the individual user, but already the earliest (hypothetical) hypermedia system, the Memex, featured in addition to support for personal trails, the ability to share trails among other Memex users. Apart from making trails/structures upon existing information, hypermedia systems have also found use in assisting the process of writing new material, where the hypermedia system is for instance used to gather and structure information, that is subsequently organised into a linear text (one example of such a system would be NoteCards [61]). From these systems, the step to support multiple authors seems natural.

NLS/Augment [40] was also in this aspect a groundbreaking system, where users could co-author material and participate in video conferences complete with cursor sharing in 1968. Later systems, such as KMS [1], also supported collaboration. In the case of KMS, collaboration was partly supported by the system itself through optimistic concurrency control, where the system would ensure that modified frames were not accidently overwritten, partly through the use culture of KMS, where most frames can be modified (or at least annotated) by the users of the system. Services such as locking or notifications were however deemed too impractical, especially considering the strict performance goals set for the KMS system.

In Intermedia [77] the notion of Webs, that is collections of links, became a useful tool for collaboration. Just as the ability to share and access the same information as in KMS is important for collaboration, the ability to work privately or in groups on a Web and then later publish it for others to see, helps to focus collaboration. Intermedia handled multiple users with different permissions, so that one person could create documents, available only for reading and linking by others. Another of the classic monolithic hypermedia systems, NoteCards, has also been extended to support collaborative authoring [94].

SEPIA [58, 59, 93] is an ambitious collaborative hypermedia system from GMD IPSI, Germany. Based on a cognitive model of the authoring process, SEPIA supports collaborative authoring. Collaboration is handled through different coupling modes ranging from uncoupled (a single author working on a document) to loosely (authors collaborating) and tightly coupled (authors seeing each others' updates as they are made) modes. Coupling modes for documents are changed automatically as more or less authors work on a document. An extension to SEPIA, DOLPHIN [72, 92] was created to support activities around (possibly distributed) meetings, such as pre-meeting planning, submittal of material to be a part of the meeting, as well as discussions, brain storming, and decision making. Collaboration could take place distributed, or local, using technologies such as LiveBoards (large touch sensitive displays) as whiteboards.

Among the open hypermedia systems, the collaborative aspects of HyperDisco has been described above (Section 4.3.4). Also the Devise Hypermedia has, in some of its incarnations [49, 50], supported collaboration. Webvise is also used in the Manufaktur

system [22, 79], a collaborative system which uses a spatial metaphor for arranging and organising documents and other computer artefacts in 3D.

Some aspects of collaboration are more easily accommodated by monolithic hypermedia systems. Most collaborative hypermedia systems have centred around authoring, not only of hypermedia structures (the Arakne Environment handles for instance only this kind of collaboration), but also documents. In monolithic systems, the documents are under the complete control of the hypermedia system, thus facilitating easier monitoring of changes, and enabling the possibility of locking parts of documents. The main approach to collaboration in the Arakne Environment, apart from sessions and shared awareness (described more fully in [15] [P4] (Section 6.4)), is the notion of collaborating through the annotation and structuring of shared artefacts, in this case Web pages. By such an approach, users can build a shared understanding of a problem area, just as they built an understanding by constructing a 3D world of documents in Manufaktur.

4.6 Hypermedia: Concluding Remarks

The field of hypermedia research encompasses much more that has been covered in the previous sections. The purpose has been to show the shift from monolithic hypermedia systems to open hypermedia systems, to discuss some of the systems and activities undertaken by people in the open hypermedia field, and finally to touch upon the collaborative aspects of hypermedia. Hypermedia research, in one form or another, is very much alive today, as witnessed by the solid participation in the Hypertext conferences. There is much interesting work being done, however in this chapter I have only been able to touch upon that which has had direct relevance upon my work.

The field of CSCW is likewise an active field. While my work has concentrated predominantly on hypermedia; some CSCW aspects, especially shared awareness, has contributed to my work.

Open hypermedia as such has been my tool of choice throughout my work. The area where I have employed this tool is the Web, and therefore a description of the Web as well as some related areas is in order.

4.7 A Brief History of the Web

The Web emerged in the beginning of the nineties [14]. It was developed by Tim Berners-Lee at the European centre for high energy physics CERN in Geneva. It was conceived as a technology that would allow visiting physicists to remain "connected" to the experiments in which they had participated at CERN, using *one* program rather than many heterogeneous programs (`telnet`, `gopher`, `ftp`, etc.). Arguably the main innovation was the Uniform Resource Locator (URL), which elegantly combined the various Internet protocols with machine names and the hierarchical Unix file system[4] as seen in Figure 4.4. By using URLs it became possible to uniquely identify anything accessible on the Internet.

Another, not as innovative but still essential, part of the WWW was the HyperText Markup Language (HTML). HTML allowed users to markup their documents to some degree and to insert URL links into these HTML pages. Finally, the WWW needed a

[4]The last part of the URL is usually based on a file system, but may in fact be any string to uniquely identify a resource.

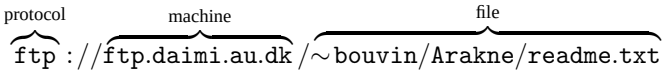


Figure 4.4: The Uniform Resource Locator

transfer protocol, i.e. the HyperText Transfer Protocol (HTTP), which at the time[5] was a simple stateless file transfer protocol. HTTP was created as the overhead involved in establishing a FTP connection was too great, especially as most Web use need many connections created and destroyed over ordinary browsing.

The Web was originally conceived as collaborative, and the first graphical Web browser developed by Berners-Lee for the NeXT was a browser/editor [13]. This browser however never gained popularity and was soon overtaken by the NCSA Mosaic browse-only browser, which since has morphed into the Netscape and Microsoft browsers.

The Mosaic browser was easy to use, ran on many platforms (the author used Mosaic extensively for the first time in CERN in 1993), and was free. "The Internet", which until then had been the realm of academics and specialists, suddenly became usable for most people and it also became easier to publish material on what now was "the Web". This effect snowballed as few things have snowballed before, and leaves us today less than a decade later with millions of Web users with access to approximately one billion Web pages, and where the average Westerner daily is bombarded with URLs for additional information and commercials. The Internet is no longer the realm of the professional — it has become a truly global information space.

What constitutes the success of the Web, that it has succeeded where other public information systems failed? Some of the more important factors contributing to the Web's success were:

- Simplicity
- Freely available standards and software
- Decentralised architecture
- Scalability
- "Real" documents[6]

These are in essence the characteristics and strengths of the Internet itself, and by leveraging these strengths, and by admittedly having the right technology at the right time, the Web became huge.

4.7.1 Hyper-G

The Web did not rise to sovereignty unchallenged. Inspired by the early Web, the researchers at Graz University in Austria created the Hyper-G (later HyperWave) system [9, 75]. Though very similar in some aspects (both used e.g. tagged ASCII almost compatible document formats), the Hyper-G system also provided some advanced functionality of its own. These included hierarchical navigational structures, and built-in multiple language support.

http://www.hyperwave.com/

[5] It has since become much more complicated — the current specification is ∼150 pages long.
[6] As opposed to Teletext or Gopher.

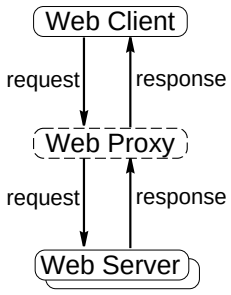


Figure 4.5: The General Web Architecture

In many ways Hyper-G was technologically superior to the Web. Hyper-G offered (automatically) indexed document collections, that could be shared and combined arbitrarily. The collections could be distributed and replicated automatically. Using the appropriate clients, users could create bidirectional point-to-point links, that were stored outside of the linked documents. Hyper-G was multi-protocol, as it could interface to Gopher and Web clients. And yet, it did not replace the Web.

Several reasons for this can be found. One of the selling arguments of Hyper-G was that it made Web sites easier to administer, as the system handled documents, and updated links etc. if documents were moved. This is certainly a valuable service, and a necessity as Web sites grow large. However, when faced with this problem, the common solution (and by far the most widespread today on the Web) was to use "real" databases to form the backend of the system. This is not a dedicated hypermedia system, but offers most of the advantages with regards to ease of administration, and has been shown to scale well. Another problem with Hyper-G was the need for proprietary clients. While Hyper-G offered a Web interface, the full benefits of the system could only be reaped through the Hyper-G clients. At the point of introduction (1995), the Netscape Navigator was the dominant browser with more (visual) bells and whistles than the Hyper-G clients. In essence, the history of Hyper-G illustrates one of the lessons of open hypermedia — that is hard to get people to cross over and abandon their favourite tools.

4.7.2 The Semantic Web?

A much heralded theme of recent Web conferences has been the "Semantic Web", a future where meta-data of various sorts will optimise the way, humans use the Web by allowing us to find what we want faster by having machines infer relationships between documents. Technologies to support this include RDF (Resource Description Framework) [70], a XML based meta data format. A RDF document basically consists of statements in the form of three-tupels, consisting of a resource (identified with a URI), a property type (an attribute of the resource, such as 'author'), and a value (such as 'John Doe'). A statement can itself be a resource for another statement, such that statements become nested. RDF is a relatively new standard and is yet to see widespread use. While not yet implemented, the RDF interest group states querying and searching as future work. To support different meta data standards, RDF supports XML Name Spaces, so that e.g. the Dublin Core can be represented in RDF. `http://purl.oclc.org/dc`

The vision of meta data and specifically RDF on the Web is that the future Web

would consist of not only Web sites with ordinary Web pages, but also collections of meta data making statements about these Web pages. This presumably rich meta data could then be used for queries and inference. A basic problem with this scenario, a problem shared with hypermedia in general, is that this meta data must be generated by someone. Some discussions have involved automatically generating meta data from Web pages, but such meta data would not contain more information than the originating Web pages. Not only must meaningful meta data be created, a difficult task in itself, but if it is to be used for machine inference, the specification schemes must be followed fairly accurately. Taking the current Web's relaxed relationship to, say, HTML standards, this seems a bit optimistic. There can be no question that meta data works excellent in e.g. library settings, where index cards etc. are produced by trusted professionals in an established fashion, but as Web search engines research shows, people in general cannot be trusted to describe their Web sites truthfully[7].

The problem of getting someone to author the meta data is, as mentioned above, also shared by hypermedia. Someone has to author the links. However, links (or other hypermedia structures) in general are *human* associations accessed by other humans, not building blocks to be used by computers, and humans are generally better to sift information out of even questionable associations.

The general semantic Web seems at this point unlikely, if nothing else, just because it expects the authors of the Web to adhere to shared encoding standards in the description of their pages. Data warehousing is the difficult field of combining the information systems used by a large company into one system. This turns out to be a very difficult problem in practice, even within just one company, given the different focus of different communities of use within the organisation. Data warehousing the meta data of the Web will be much, much worse.

4.7.3 Collaboration on the Web

Due to its pliable architecture, the Web forms the infrastructure for many kinds of applications, not least the support for collaboration. Collaboration on the Web takes many forms — from classical CSCW applications such as BSCW [12] to link recommendation and discussion forums such as Slashdot . The former is a system to support document sharing and authoring, as well as discussions. The latter has become quite successful and has spawned many similar sites, as the software driving the site is open source. The readers of the site contribute links to interesting articles or other newsworthy items on the Web; the editors of the site publish some of these link on the front page of the site, and the news item then forms the starting point for an (at times) animated discussion between the readers. To help improve the quality of the discussion, the postings by readers are scored by moderators. Moderators are not editors per se — they are ordinary readers that have previously posted good material on the boards (and have thus received a high score from other moderators). To keep the moderation fair, the rest of the readers can rate the moderation itself, thus directing the work of the moderators (of which there are several thousands). There are two advantages to these kinds of sites, that greatly increases the usability of the Web. The sites are usually fairly specialised in scope, and given the large readership (and hence the large number of contributions), a site summarises most of the interesting events within a particular

`http://slashdot.org/`

[7] Panel on search engines at Digital Libraries 1998 in Pittsburgh, USA. HTML already supports meta data in the `<head>` tag, but this is, according to the panel, largely ignored by search engines. The problem is that this meta data is often used for "keyword spamming", where the meta data field is filled by words unrelated to the page itself in the hope of getting a higher ranking in (perhaps quite unrelated) search results.

field. Furthermore the discussions following the news item will often provide insights and pointers otherwise difficult to obtain. In short, these sites allow the individual user to benefit from the collective knowledge, Web browsing, and standards of the community. Some sites, such as kuro5hin have even done away with the editor concept, so that the selection of newsworthy items among the contributions is carried out by the readers. By the readers, and for the readers.

http://www.kuro5hin.org/

These kinds of Web sites highlight what is possible with the pliable architecture of the Web. While well suited for discussions, they do not address other collaborative efforts, such as authoring.

4.7.4 WebDAV

Another example of support for collaborative work (and one closer related to the CSCW field) is Web Distributed Authoring and Versioning, WebDAV [98]. As mentioned in section 4.7, the Web was originally by Tim Berners-Lee conceived as a read/write environment, where the interaction with the Web took place in a browser-editor. To support this functionality, the HTTP protocol has long provided the `put` operation. Unfortunately, this has not found widespread use — the author is aware of but one editor-browser, Amaya, which uses it, and hence Web server support is also scarce. One problem with `put` is that there is no provisions against overwrites, as documents cannot be locked, and given the arbitrary large size of Web pages, an optimistic concurrency approach as in KMS [1] seems overly optimistic. WebDAV addresses this by extending the HTTP protocol, and introducing locks on Web resources. Furthermore it is also possible to store information about resources (or more precisely, about URIs). Versioning is not (yet) a part of the WebDAV standard, but the ΔV [36] group is expected to finish their work in year 2000. Compared to more ambitious collaborative authoring systems, the WebDAV standard is somewhat modest, as it is only possible to lock whole resources, not parts of a resource. Rather than a weakness, this can actually be seen as a strength — WebDAV is format agnostic, and can thus be much simpler to implement than a system that has to be able to handle the internals of diverse document formats, and presumably be updated as new document formats are released.

http://www.webdav.org/

http://www.w3.org/Amaya/

Apart from direct collaborative authoring of documents, WebDAV enabled Web servers opens some interesting possibilities as remote repositories. The Webvise client [54] is able to export hypermedia structures into interchange files and store these on WebDAV enabled Web servers. These files can then easily be accessed by anybody with Web access.

4.8 WWW, or What is Wrong with the Web?

The Web is hugely successful and must clearly be doing something right, as has been described in Section 4.7. However the Web could be more than it has become, and this section outlines some of the shortcomings of the current Web.

The greatest immediate shortcoming of the Web from a hypermedia standpoint is the linking model. Links on the Web have a number of constraining characteristics:

- Inline
- Author only linking
- Unidirectional

- one-ary
- Untyped
- One set of links per document
- URLs are physical adresses, rather than symbolic names

The main problem is that links are inlined in HTML documents, and this is the cause of most of the other problems. The Web link is in essence little more than a goto or a jump instruction to the Web browser to retrieve and display a new document (and in this sense quite similar to KMS [1]). The owner of a Web page is the only one who can create links or named regions in a page. As links are unidirectional, there is no way of telling whether there are links pointing to a given page or resource[8]. It is not possible to have more than one destination per link, nor can one distinguish between different kinds of links[9], such as annotations, references, etc. With the `target` attribute in the `<a` tag, it became possible to signify some link behaviour as the destination document can be displayed in an existing frame or in a new window, but it is e.g. not possible to let a link be replaced by the document that it points to. Some of these points are minor, but the perhaps most serious consequence of the inline linking model is that there can be only one set of links per document. This is a drawback as it limits the possible uses of a Web page. If it was possible to have multiple sets of links to a given page, this page could be reused in other contexts without any modifications to the page. It would e.g. be possible to publish a technical specification, the specification with annotations, the specification with links to source code implementing it, etc., without changing the specification document into multiple (slightly different) copies. Ignoring for a moment temporarily unavailable Web servers, link rot is a reoccuring annoyance on the Web. This often happens when authors or maintainers of Web pages decide to reorganise or move their Web pages. This will break links relying on the previous organisation. Partly, this problem also can be blamed on the inline link (it certainly does not lessen the problem). Maintainers of Web pages have no way of knowing or anticipating the use that others put their pages to. Thus, maintainers cannot know if moving a document will have an impact on other Web pages. The arbitrary reorganisation of Web pages would be a smaller problem, if URLs were not used as *location* specifiers — the URL has been likened to finding a book in a large library with directions like “twentieth book on the third shelf on the seventh row on the first floor”. If the books are moved around, the desired book can no longer be located. Similarly, if a file is moved to a new directory, a URL previously pointing to it becomes useless.

The Web is a monolithic hypermedia system, and could therefore be expected to be criticised for the problems associated with other monolithic hypermedia systems, such as closedness, the lack of third-party application etc. In the case of the Web, this critique does not have much merit, as the Web has succeed, where the other systems did not. The computing world has become Web enabled, and many program are able to work with the Web one way or another.

It should be made clear, that while the Web has a problematic linking model, this simple model is also what has made the huge scale of the Web possible. While certainly not perfect, it works surprisingly well, but as previous sections has outlined, there are scenarios where more elaborate structuring mechanisms become desirable. Thus, for many purposes, the Web suffices, and for the rest Web augmentation can play a part.

[8] Save by using a brute force method such as indexing *all* Web pages.

[9] It should be noted that the current HTML specification actually allows for some limited link typing, but it is under specified and has not, to the author’s knowledge, been implemented in any Web browser.

4.9 Summary

This chapter has provided a non-exhaustive overview of the fields, I have based my work on. Largely, my project of Web augmentation can be seen as extending the experiences of open hypermedia to the Web. The hypermedia research community has in its more than fifty years long history developed many exciting and useful ideas, systems and structuring mechanisms, and it would be a pity if this was not made available for the Web. The Web itself is a fascinating entity with its simple and immensely flexible architecture and the many different related technologies, many of which are still emerging. Apart from the systems already described, there are other systems, described in the following chapter, that seek to augment the functionality of the Web.

Chapter 5

Related Work

In [17] [P1], I described many Web augmentation tools. In the interim some new tools and technologies have emerged or matured, and this chapter describes some of them. Broadly, they can be divided into two groups: XLink (and related technologies) and personal tools aimed squarely at the Internet Explorer.

5.1 XLink

XLink[1] [37] is a W3C initiative to extend the linking capabilities of the Web. XLink is a general XML format to describe navigational hypermedia, and to allow expressions of navigational hypermedia to be inserted into XML documents. While the main application of XLink is expected to be linking within XML documents, the standard itself is not limited to address solely XML locations, provided that appropriate locators have been defined. XLink can support the linking currently found on the Web (e.g. unidirectional one-ary untyped links), as well as bi-directional n-ary typed links. Links can be stored externally ("out-of-line") of the documents, they address, or they can be in-line (as with HTML documents). Traversal of a link may result in replacing the document currently viewed (as is the standard behaviour in the context of HTML), or by inserting the target for the link in the viewed document. The traversal may be initiated by the user (e.g. by clicking on a link), or at the time of document retrieval.

XPointer [30] is used to identify regions of interest in XML documents. XPointer allows for selection based on ids, hierarchical structure (from XPath [27]), or an arbitrary user selection (e.g. selecting a string in the rendered XML document). This is a quite sophisticated addressing scheme that should cover most uses. XPointer can address arbitrary XML documents, but the explicit support for XPointer can also be added to a DTD. A given region may be identified using several locators, which improves reliability, as one locator might fail after a document has been edited.

XLink holds great potential. At this point it is still too early to discern whether it will hold, but *if* it does, it holds interesting prospects for the future of the Web. In a realised XLink future, complex linking structures are routinely stored outside of Web resources, immediately available for the users of the Web. In such a scenario, there will of course still be open hypermedia research (operating system research did not cease because of Microsoft Windows). Indeed, just as the Web made "hypermedia"

[1] Parts of this section is taken from my contribution to [55].

a widely understood term, XLink may have the same result on externally stored hypermedia structures. XLink is a very new technology, and so far there are only a few systems providing XLink functionality. Presumably this can be expected to change. As it currently stands, XLink is predominately biased towards navigational hypermedia[2], and while much hypermedia can be modelled in terms of links it has long since been established at least in the OHSWG community that this is unnecessarily cumbersome for hypermedia constructs such as guided tours, compositional, taxonomic, or spatial hypermedia. Still, XLink is a good starting point. Currently XLink is mainly an interchange format — there are no standards for interaction between XLink applications or XLink queries. Through XPointer and XPath, XLink is supremely well suited for linking XML documents. It is however completely feasible to write locator (or LocSpec in OHSWG parlance) specifications for non XML documents, e.g. images [64].

5.2 Internet Explorer Add-ons

A new form of Web augmentation tools has emerged since [17] [P1] — the specialised Internet Explorer add-ons. The Internet Explorer lends itself supremely well to this kind of integration, as it is a COM (Component Object Model) component and has an excellent interface[3] for accessing the Document Object Model of the displayed document. Functionally most of these new tools integrate themselves directly into the Internet Explorer interface by added a new tool bar or menu.

The tools described in this section are commercial, and as such have to generate revenue for their companies. The approaches taken by their respective owners are interesting and in some ways similar. Flyswat and Third Voice both generate what are essential generic links by highlighting key phrases in the displayed document. These links will typically contain links to sites related to the phrase and, noteworthy in this context, commercial ventures — e.g. the phrase “Pokémon” or “Pikachu” will have links to sites where this merchandise may be purchased. Flyswat directly offers partnerships with companies, so that they can make their services directly available to users through the Flyswat service. Both the Flyswat and the Third Voice clients are free, and presumably generate their revenue from the “product placement” partnerships. In contrast, iMarkup, which seems to have a different marketing plan, currently charges money for their clients and their servers.

This is perhaps not exactly what Bush had in mind, when he wrote of trail blazers. It seems, at least to the hopeful author, that there may be room for both kinds of activity on the Web. The in some circles much lamented commercialisation of the Internet has in retrospect had many benefits, as it by the economy of scale has made the Internet available to the “non-technical” population. Just as the Web made hypermedia accessible to everybody[4], these kind of tools can perpetrate the idea of adding another layer of information on top of existing Web pages. iMarkup is in this context interesting as their aim, apart from the individual markup of pages for personal use, is to provide groups and organisations with shared annotations. This is an daring endeavour, and whether their business plan will translate in profitability remains to be seen. As such

[2] It should be noted however that according to Steve DeRose and Lloyd Rutledge (both participants in the XLink authoring process) that the term “link” in the XLink standard document should be read as “relationship” (private communication).

[3] This interface is also the reason why the Arakne Environment at an early date changed from the Netscape Communicator to the Microsoft Internet Explorer. The difference in interface richness is dramatic.

[4] In the so called Western World, that is. The “digital divide” between the Western World and the rest of the Earth remains very much in place.

these companies' fate can be quite crucial as their products are not too dissimilar from "real" open hypermedia systems integrating the Web, or indeed open hypermedia in general. If they succeed, so may we. One item of concern is that iMarkup and Flyswat both announce on their Web sites that they have secured or are in the process of securing patents on their respective "key technologies", though exactly what entails remains unclear. Whether this will have implications for hypermedia research is an open question.

5.2.1 Flyswat

Flyswat is an ActiveX component integrated with the Windows operating system. When enabled, it highlights keywords on Web pages displayed in the Internet Explorer, so that a click on the keywords can lead to a definition of the word, the Web site associated with the term, or a page where the item may be purchased. The Flyswat functionality is not limited to the Internet Explorer (though markup is found only in the Internet Explorer) — a Alt-Click on any word in a Windows application will result in a lookup. The effectiveness of Flyswat greatly depends on whether a given subject is covered by the Flyswat company or any of the associated partners. There does not seem to be any possibility of submitting links or keywords to the service, outside of becoming an associated partner with Flyswat.

`http://www.flyswat.com/`

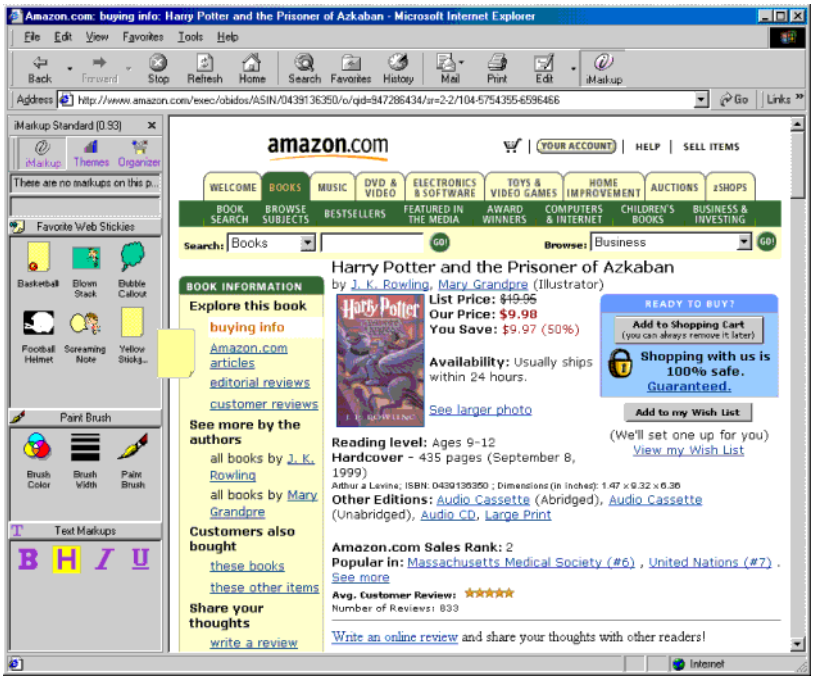


Figure 5.1: The iMarkup Application (from `http://www.imarkup.com/`)

5.2.2 iMarkup

http://www.imarkup.com/

iMarkup is another add-on to the Microsoft Internet Explorer. It provides the user with the ability to interactively markup Web pages with highlighting, annotations, and freehand drawings. The marked up pages are stored locally, but can be sent to others using email or ICQ (a popular Internet instant messaging service). iMarkup is closely integrated into the Internet Explorer's interface, as seen in Figure 5.1, where iMarkup resides in the leftside of the window.

The iMarkup is now also available distributed. By adding an iMarkup Group server to a Microsoft IIS Web server, groups can access shared annotations. The server is a commercial product and the license fee depends on the number of users.

Given its proprietary nature, it is unclear from the company's documentation, how the markup is achieved and how markup is localised in conjunction with the rest of a Web page. According to the company's Web site, iMarkup employs patent pending techniques. However, it is most likely that the system utilises the same techniques as Webvise and Arakne Environment, that is accessing and manipulating the DOM (Document Object Model, a hierarchal model of a Web page) of a Web pages to add new content.

5.2.3 Third Voice

http://www.thirdvoice.com/

Third Voice is an add-on to the Microsoft Internet Explorer. Originally, Third Voice offered users the ability to annotate Web pages with "sticky notes", very much like iMarkup. However in the latest version, this has changed radically. Third Voice now offers services similar to Flyswat, including the partnership model. Additionally users of the service can discuss Web pages or topics in boards provided by Third Voice.

5.3 Summary

The different directions taken by the above described systems are interesting. XLink is an attempt to provide good general navigational hypermedia for especially but not only XML documents. It takes its outset in defining open standards, which developers are free to adopt. Once the different standards have settled, implementations will hopefully follow. At the other side of the spectrum, we find commercial tools, which offer services today, but does so with closed, indeed sometimes "patent pending", standards. I can only wish these companies luck in their bold ventures, but from an open hypermedia standpoint, I hope that open technologies such as XLink will prevail, or that these companies eventually will make their systems open for others.

Chapter 6

Contributions

This chapter presents the main results of my Ph.D. project. My method of work has been that of experimental computer science, where theories and concepts are validated through implementation of prototypes. My starting point has been the concept of Web augmentation, and through theory building of conceptual frameworks and the implementation of prototypes I have examined this topic. This work is reflected in the following papers.

6.1 [P1]: Unifying Strategies for Web Augmentation

Web augmentation tools has been a recurring theme of this text. Such tools are available in many different forms, and the Arakne Framework seen in Figure 6.1 is an attempt to unify the various approaches taken by these tools. This has been documented in [17] [P1], presented at Hypertext 1999 in Darmstadt, Germany.

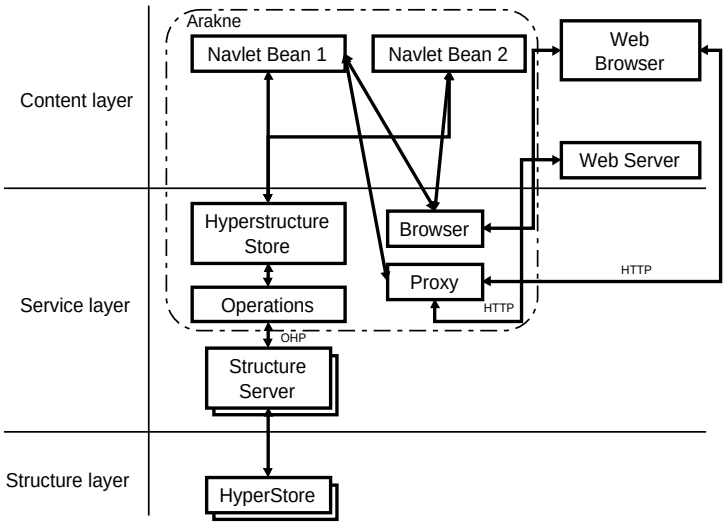


Figure 6.1: The Arakne Framework

One of the basic problems of Web augmentation is to maintain control over the Web pages and the Web browser as described in Section 4.3.9. In general Web augmentation tool developers wish their tool be remain active and relevant to what the user is doing with the Web. Thus, the system must be kept aware of what Web page is being displayed, and may also need some way of modifying the contents of the current Web page. These seemingly simple requirements have not been without some difficulties during the progression of the Web and the dominant Web browsers.

The Arakne Framework can be seen in Figure 6.1. It was based on the analysis of many, quite different, Web augmentation tools, and models such tools by identifying the necessary constituent components and their interactions. A Web augmentation tool needs an interface so the user can interact with it; it must know which Web page is being displayed; it needs to retrieve the information pertinent to the Web page in question; and if such information exists, it may need to modify the Web page. These tasks are reflected in the Arakne Framework. The Arakne Framework can be used to model arbitrary Web augmentation tools, and some examples of this are given in the paper.

The Arakne Environment, presented for the first time in [17] [P1], followed the Arakne Framework. One of the purposes of the Arakne Environment has been to implement the components used by Web augmentation tools, so that prospective developers can concentrate on the more interesting task of developing their tool's core functionality. As such, the Arakne Environment supports an open set of Web augmentation tools. The development leading to the current version of the Arakne Environment is described in more detail in Section 6.6.3 and Chapter 7.

6.2 [P2]: Opening Temporal Media for Web Augmentation

The Web is more than HTML pages. As the common Internet connection becomes faster, other media types, such as audio or video, become more widespread. In order to demonstrate that the open hypermedia Web augmentation approach could be extended to more than HTML, the Coconut project set out to create a hypermedia integration of video files with Arakne. This work has been documented in [20] [P2], presented at WWW9 in Toronto, Canada.

At the time of development, we were faced with the recurring problem of open hypermedia: Content handlers that could not be sufficiently integrated for our purposes. We could do "launch only" integrations, but this is also possible with ordinary Web links, and would thus be a rather pointless exercise. We therefore resolved to create our own media player, which was to mimic the media player used by the Web browser, hence the name Mimicry. While not trivial, the task of creating this media player was greatly helped by the Java Media Framework, which is able to handle a rich set of time-based media types. At the point of creation, the Arakne Environment did not do its own link decoration, as we had yet to implement the full Internet Explorer integration. Web page modifications were handled by the DHMProxy [56], which in this case identified embedded media clips (identified by `<embed>` or `<object>` tags) of the types handled by Mimicry, and rewrote these to `<applet>` tags, invoking the Mimicry applet to play the media file, as well as specifying the anchors, if any, in the media clip. Through this technique, the normal media players or plug-ins were circumvented, and we had a media player with a much richer programming interface. Through integration with the navigational view Navette in the Arakne Environment, it was thus possible for users to

`http://java.sun.com/products/java-media/jmf/`

author links with anchors in ordinary HTML pages as well as in media clips as seen in Figure 6.2. The general architecture of the Arakne Environment at this time can be seen in Figure 6.3.

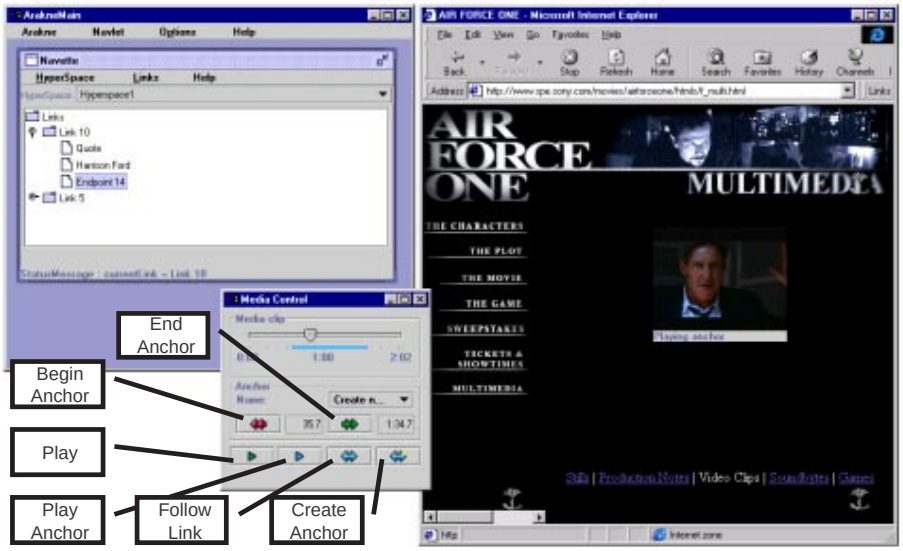


Figure 6.2: Mimicry playing the endpoint 'Endpoint 14'

Mimicry was largely an experiment, and has not received much development since. At the point of development, the Java Media Framework did not support streaming media. This has been corrected in later releases. Today, the situation is however slightly different, as at least the Microsoft Media Player has a sufficient interface to support open hypermedia needs (and has e.g. been integrated with Webvise), so the need for our own media player may no longer exist. A problem of the temporal media types is that they often rely on proprietary formats and players, which makes an integration more difficult. This is of course nothing new in the context of open hypermedia.

Apart from illustrating that Web augmentation is not limited to HTML pages, the main contribution of the paper is the identification of the problematic closed nature of plug-ins and other media handlers. Not only does this make life more difficult for open hypermedia people (admittedly a minor concern for the developers of these programs), it also makes it more difficult for "normal" Web authors to utilise such media in creative ways.

6.3 [P3]: The Arakne Environment and the future of OHSWG

The experience of porting Arakne to Construct, or more generally to OHP is documented in [19] [P3].

The Arakne Environment originally relied on the DHM hypermedia servers. As the Coconut project progressed, and the Construct servers matured, it was a natural step to

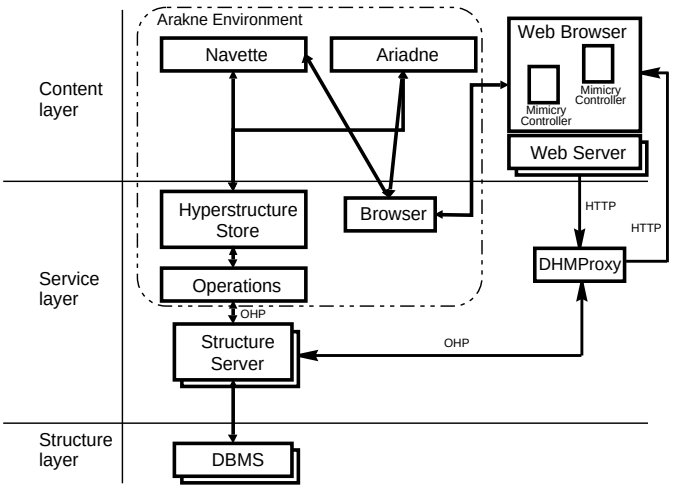


Figure 6.3: The Arakne Environment with Mimicry

move from DHM to Construct. This would provide Construct with a client for testing purposes, and provide the Arakne Environment with the richer functionality, especially with regards to collaboration, of Construct.

The move to Construct was accomplished in two steps. Version 2.0 of Arakne was the first to utilise Construct, and did so by essentially wrapping Construct. This was done for two main reasons: Firstly to make a speedy transition, and secondly to isolate the views from the changes made in the backend.

At the time of this first step, the Arakne Environment was organised as seen in Figure 6.3. While the Internet Explorer could be controlled from the Arakne Environment (for purposes such as reloading pages), link decoration was still handled by the DHM-Proxy. The views in the Arakne Environment provided the GUI for creation of links and guided tours, and interfaced to the "Hyperstructure Store"[1] for the required functionality to create these structures at the hypermedia server. As one of the priorities of the first transition was not to break existing views, the interface from the Hyperstructure Store to the views was retained. To fit within Construct, the Hyperstructure Store and its associated data model was drastically changed. The data model was changed, so that each class (such as Link or Anchor) became wrapper classes for the matching Construct classes. While tedious work, this was fairly simple, especially as Construct like Arakne is written in Java. While this first integration made the creation of links and guided tours possible, the more advanced features of Construct, such as collaboration support through sessions was still not available. We therefore resolved to make Arakne a native Construct client. This required a more elaborate redesign to accommodate the Construct data model and to handle sessions correctly. This new architecture, which is the current one, is described in more detail in Section 6.6.3.

Having ported Arakne to OHP, it was time to take a step back, and consider OHP. It has many good points, and is a very advanced hypermedia architecture. It is also an open standard, yet there has to this date been zero migration to it outside of OHSWG. This led to some considerations about OHSWG and the nature of standardisation.

[1]Note: Store as in "convenience store", not as in "storage".

Over the years, OHSWG has held demos at the Hypertext conferences to demonstrate the advances made by the group with regards to interoperation and collaboration support. Up to these events the participating parties has more often than not been feverishly working to correlate their almost but not quite compatible implementations of OHP. Thus, even within OHSWG itself, the OHP standard is not clear. OHP is currently defined de rigueur if not de facto by the "Darmstadt DTD", which specifies the protocol and data model in an XML DTD (Document Type Definition). DTDs are suited for specifying syntax, but not semantics. Thus, a lot of the specification is not in the DTD, but is essential left to the implementor. Currently the only complete implementation of OHP is Construct, and there has invariably during the implementation of this architecture been made deviations from the OHP standard. This is to be expected, as implementation is one of the better ways to evaluate standards. However, for future purposes and for the cause of OHSWG, one implementation of OHP is not enough. It is unlikely that Construct can cater to all needs and it is more than likely that idiosyncracies of the Construct will remain undetected unless they are verified against an independent implementation. So, if the goal is to make OHP more widely accepted and employed, we must consider the means.

One obvious lack of OHP today is that it is underspecified. One DTD does not a standard make, and given that Construct demonstrates that OHP can survive implementation, the time has come to create a more substantial standards documentation. To this end, a DTD is unsuitable, because it is too unclear on semantics, while being too low level. A good specification should not tie itself to one transport protocol, such as XML over sockets. I participated in decision to recommended that XML should be used over sockets for the entry-level implementations, while CORBA should be used for more ambitious implementations. This decision was perhaps in retrospect suspect, but it made sense at the time. XML was widely supported on many language platforms, it was easy to debug, as it was human readable, and it would lend itself readily to e.g. an interchange format. A major consideration was also the decision not to bind ourselves to one particular platform (language or otherwise), such as DCOM which even today is predominantly a Microsoft platform, or CORBA, which at the time was not freely available. The downside to XML became clearer as the various implementations progressed, as there were no tools to unambiguously generate skeleton source code from a specification, as is possible with e.g. IDL (Interface Description Language). This led to the problems with interoperability described above. Today the situation has changed. CORBA is now more widely supported, and another interesting development is the CSC tools developed at Aalborg University Esbjerg [105]. Based on an IDL or UML (Unified Modelling Language) specification, these tools generate skeleton code for Construct services. The transport layer is independent of the IDL specification, which rightfully becomes the main focus. These tools are a step in the right direction. Future work for the OHSWG will be to create a proper standards based on higher level transport agnostic specifications of both data model and semantics with a clear mapping from specification to implementation. CSC provides one such mapping, and I have volunteered to coordinate the writing of an OHP standard — a task I expect to commence this autumn.

The contributions of this paper are firstly an discussion of how a hypermedia client can be ported from one hypermedia system to another with special focus on the tradeoffs involved. Secondly, it also contributes to the on-going discussion within the OHSWG to further the state of open hypermedia.

6.4 [P4]: The Collaborative Arakne Environment

Support for collaborative hypermedia authoring became a priority in Arakne Environment with the introduction of the Construct servers. The work to support collaborative work in Arakne is documented in [18], presented at Hypertext 2000 in San Antonio, USA and in the yet unpublished [15] [P4].

For a more general discussion of CSCW, see Section 4.5. The basic notion of collaboration in the Arakne Environment was present before the move to the Construct servers (see Section 4.3.7), which supports collaboration explicitly. The previous versions still allowed people collaborate implicitly through the annotation of Web pages with notes or links. With the move to the Construct servers, it became possible to take this collaboration to Arakne itself, and how this was accomplished is the topic of this paper.

OHP and thus Construct handles collaboration through sessions. A session is defined by a coupling mode, a set of users, a set of tools, and a set of documents. Of these the coupling mode is the most interesting. The coupling mode defines the visibility of each users' actions. The current version of the Arakne Environment supports three coupling modes: Uncoupled, loosely, and tightly. The difference between these coupling modes lies in the frequency with which updates are broadcast to other members of the session. In the uncoupled mode, there is no broadcast at all, and in the tightly coupled mode, all changes are broadcast.

To supplement the coupling modes, OHP also supports subscriptions on OHP events. This allows a user to for instance be alerted, when John creates a link, but not when Jim does.

To accommodate this new functionality without disrupting the existing user interface too much, an attempt was taken to make it possible to collaborate without having to focus on the collaborative parts of the interface, such as the Session Manager. This was accomplished to a degree, where a user need only use the Session Manager for creating or deleting sessions, but where the switch between active sessions or the awareness of other users online has been integrated into the user interface. The main interface for keeping users aware of each others' actions is a ticker tape, located at the bottom of the Arakne Environment (visible in Figure 6.8). This ticker tape displays items such as the subscriptions described above, as well as monitors users logging in or out of the system. It should be noted, that at the time this paper was written (Spring 2000), the Arakne Environment had an Internet Relay Chat view, which allowed participants in a session to chat together. Due to problems with the IRC protocol supported by the libraries used by this view, it is currently not available, though it can be expected to be reinstated, once these bugs have been fixed. Given the automatic upgrade functionality of the Arakne Environment, such a change will be seamless to the users.

The contributions of this paper is a discussion of how the abovementioned goals may be achieved, and how one may use a collaborative framework as the one found in Construct to facilitate shared awareness.

6.5 [P5]: The iScent Framework

As it stands today, the Arakne Environment sports a nucleus of message notification through the use of a ticker tape. However, much is still lacking before such a message notification system can be regarded as truly useful. Firstly, the range of the Arakne ticker tape is limited to events taking place within connected OHP compliant clients

— typically other Arakne Environments. Secondly, the information provided in the messages is not very detailed. Thirdly, if a user misses a message as it scrolls by, it is lost for good (the object the message was about might still be present, but how would the user know?). Fourthly, while compact and easy to handle, the ticker tape as a tool is fairly limited in what it can actually do (despite the successes of Elvin [41] ticker tape).

The original goal of the Arakne ticker tape was to help users stay aware of each others' actions, that is to maintain a shared awareness (for a further description of systems supporting shared awareness, see Section 4.5). To further shared awareness[2], to address the concerns raised in the previous paragraph, and to design a versatile architecture using a message notification system as infrastructure, Kenneth Anderson and I have devised the iScent framework. This work is to be published in [6] [P5].

There are numerous message notification systems available today, both commercially and described in the literature (e.g. NSTP [85]). The beauty of these systems are the publish/subscribe model, where parties can subscribe to message matching certain criteria, without having to specifying the origin of the desired data. Similarly, other parties can publish information on the system, knowing that interested parties will get the data, but without having to establish direct connections with these parties. All routing of messages back and forth between parties is handled by the transport layer, that is the message notification system.

The iScent framework does not attempt to replace existing message notification systems. Rather, it utilises a message notification system (currently Siena [26]) as infrastructure or backbone. The purpose of the iScent (**i**nter**s**ubjective **c**ollaborative **e**vent e**n**vironmen**t**) framework is to help people increase intersubjectivity and to augment their (collective) memory through the use of high fidelity persistent events. In the extreme case all tools used by a user would be iScent enabled, and then all actions taken by the user with the computer would generate iScent events. These events would be stored by sinks, and could later be queried and retrieved by the user. The general architecture of iScent can be seen in Figure 6.4. The two kinds of arrows to the Arakne Environment illustrates that it is both an iScent aware application (i.e. generates events) and contains the Trail Viewer for visualising trails of events.

The iScent framework is still a work in progress. While the general architecture and protocols are by now stable, the greatest challenge and greatest benefit will be the development of a powerful client (the "Trail Viewer"), able to allow users to visualise and structure events, as they see fit.

A novel part of the iScent framework is the introduction of sinks. Rather than letting events and messages be fleeting things, the sinks store *all* iScent events. Sinks can through subscriptions be setup to store specific events (such that a working group would utilise a local sink rather than relying on one further away). At a later point, a sink can be queried to return matching iScent events. Off hand, this sounds like the ultimate Big Brother scenario. Why is this useful? There are several scenarios, where such a system comes in handy:

Augmented memory Jane recalls she was using a really good resource, when she was working on the Epsilon project, but where was it? Looking in her calendar to pinpoint when she was involved in the Epsilon project, Jane queries "DocumentOpened" events in this interval in the Trail Viewer, and quickly locates the relevant document.

[2] Or rather reflected awareness (also known as *intersubjectivity*)

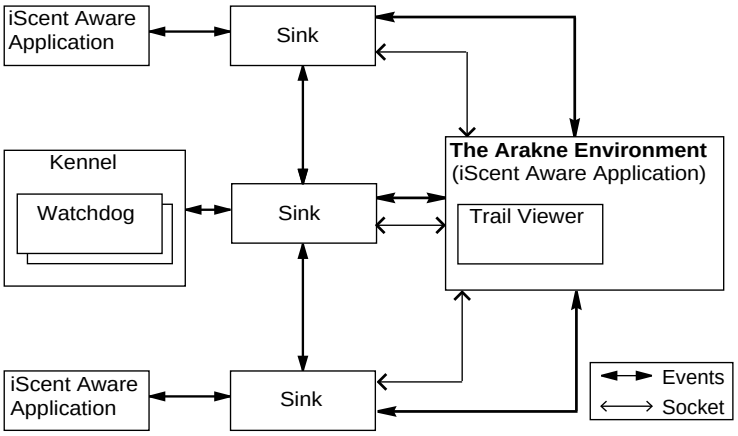


Figure 6.4: The iScent Framework

Discovering relations John is working on the database part of the Phi project, and while reading some online Oracle documentation, queries the Trail Viewer to see who else has read that particular page. Apparently Jane read the same page some months previously, and John sends Jane a technical question in an email.

Intersubjectivity Jane is writing some documentation, and would like some feedback from John. After finishing the page, she sticks a watchdog on it, so that she will be alerted, whenever John reads the page. She then emails John to tell him about the page. Later John gets around to reading the page, and by doing so sets off the watchdog. He is himself alerted to this, as is Jane. Jane now knows that John has read the page (and John knows he can expect a call from Jane), and can thus call him to hear his opinion.

With regards to privacy concerns, it is of course crucial that the individual can control what and when to publish events, and this can easily be accommodated in the client interfaces. As described, the iScent framework could lend itself to abuse by being used for monitoring, and a model of use, where events belonged to and were controlled by the originator, will have to be implemented. It should be noted that while some actions may be judged private, such as Web browsing, others such as committing a file into a CVS repository are public actions. As mentioned in Section 8.3.1, one approach to ensure privacy would be to encrypt all events at the originating machine using the user's personal key or keys belonging to projects, in which the user participate.

The main contribution of this paper is the concept of supporting intersubjectivity through persistent events and triggers. This is a work still in progress, but the general architecture of iScent seems sound, and could be used in many settings.

6.6 Prototypes

Throughout my Ph.D. I have designed and developed several Web augmentation prototypes. This chapter will describe them in turn, briefly beginning with DHM/WWW[3], with a special focus on their design, and how the experiences from one design iteration helped create the next versions. As is the fate of prototypes, these systems have been abandoned as new versions were created, so only the latest version, the Arakne Environment 2.1 is publicly available at the author's Web site .

`http://www.bouvin.net/Arakne/`

The experiences gained and difficulties encountered during development with regards to Web browsers and Java are described in Section 7.1 and Section 7.2. This section concentrates on the functionality of the prototypes, and the rationale behind the design.

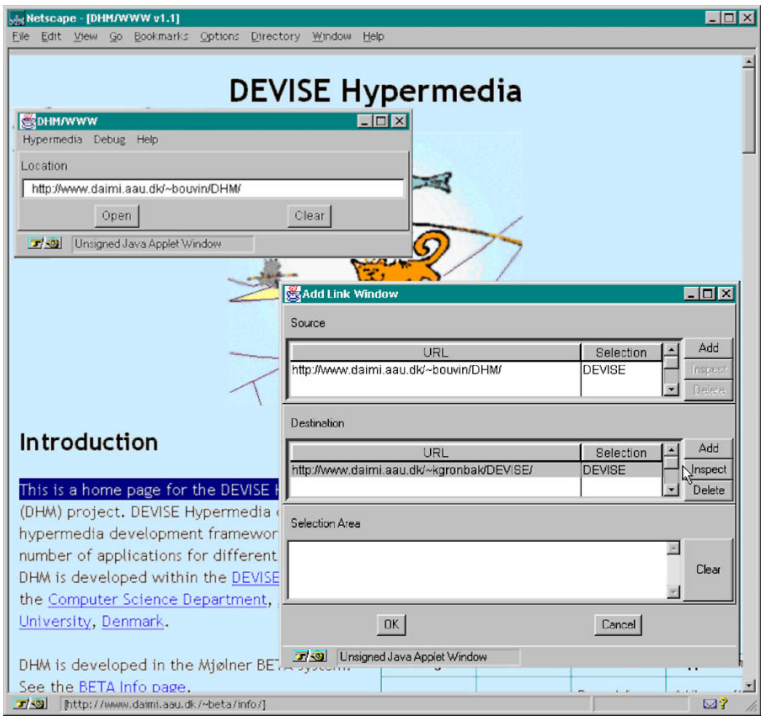


Figure 6.5: The DHM/WWW Prototype

6.6.1 DHM/WWW

DHM/WWW [48], shown in Figure 6.5, was a prototype to showcase the possibilities of applying open hypermedia to the Web. Functionally, it was a Java applet, and as such quite easy for the user to start — this only required the user to retrieve a Web page containing the applet. Once the applet had started, the user could then proceed to browse Web pages as before, provided that the user constrained the browsing activities to the Web server whence the applet came, did not use bookmarks, and avoided pages

[3]Strictly speaking, this system is prior to my Ph.D., but it set the course for my work, so I decided to include it.

with frames. Under these restrictions, the user could create bi-directional *n*-ary links, as well as browse existing links. Link creation involved copying and pasting a context around the desired endpoint into the selection area and then selecting the endpoint. There was only one collection or hyperspace of links, and no notion of different users. Link decoration was pre-render, and thus fairly slow. The applet ran on all platforms supported by the Netscape Navigator, and used CGI-scripts to communicate with the DHM hypermedia server.

6.6.2 Navette

Navette [16], shown in Figure 6.6, was the first prototype, I created during my Ph.D. It was, just as DHM/WWW, a Java applet. Where DHM/WWW had been in separate window, Navette was placed in a frame in the Web browser, as the small DHM/WWW window tended to disappear under other windows. To address the limitations of the "Java Sandbox", Navette was digitally signed, allowing it to access arbitrary Web and hypermedia servers. To increase performance, communication with the hypermedia server was handled directly through TCP/IP. Link decoration was handled through a proxy thread, as the Netscape browser supported dynamic proxy configuration changes through digitally signed Javascript. The newly introduced event model reduced the complexity of Web page modification considerably by eliminating the need to modify every link, and this coupled with the proxy made link decoration considerably swifter, if not as swift as hoped for. Navette supported multiple users (through not collaboration as such) and collections of links. While the event of a user clicking on a link ("`onClick`") could be detected across frame boundaries, and hence by Navette, the act of selecting text ("`onSelection`") could not, regardless of digital signing, so link creation was still a two-step process. Interface wise, the GUI had been broken up into tabbed panes to maximise the space allotted to its frame. While more robust than DHM/WWW, Navette was still susceptible to Web pages with frame sets (because these frame sets could overwrite the frame Navette resided in, thus terminating the applet), and the use of bookmarks.

6.6.3 The Arakne Environment

Arakne was my third Web augmentation prototype. Arakne was originally created to accommodate the two different Java Web augmentation tools developed by the Coconut project members, Ariadne [66] and Navette. Rather than adding the functionality of one to the other, it seemed more promising to create a system wherein the tools could co-exist. There were shared functionality between Ariadne and Navette: Both needed to communicate with a DHM hypermedia server, and both required some control over a Web browser, so this functionality was factored out. The earliest known screen shot of Arakne is seen in Figure 6.7. The screen shot is unique in the sense, that Arakne evidently at this point still was an applet. Soon after, the applet idea was abandoned, and Arakne became an application. There were several reasons for this. Firstly, applications do not have any of the security restrictions imposed on applets; secondly, applications are not as fragile as applets, i.e. applications do not cease to exist, just because the user retrieves a new Web page; thirdly, the work on the local proxy had ceased, and instead Arakne employed the DHMProxy [56] for link decoration, removing the link decoration need to tightly manipulate the Web browser. At this point, we switched from the Netscape Communicator to the Microsoft Internet Explorer. At this point, the Internet Explorer had become the dominant Web browser, and it had a rich

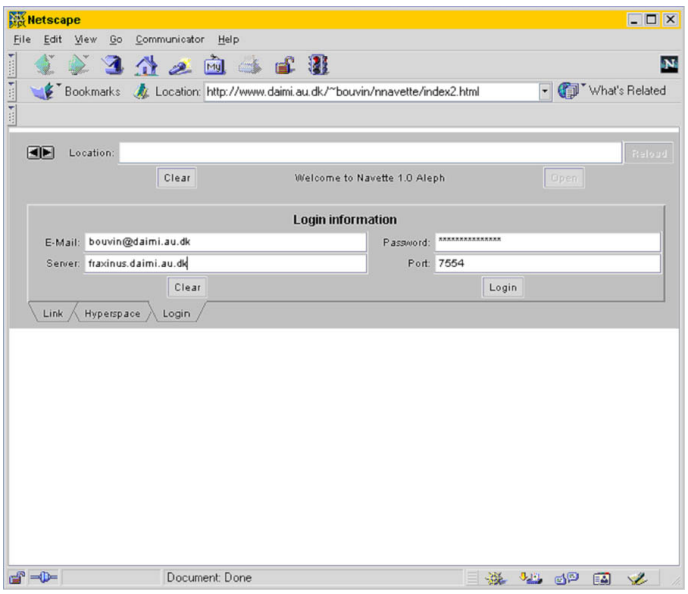


Figure 6.6: The Navette Prototype

interface for controlling the browser and manipulating the displayed Web page. Arakne ran on Microsoft Java, which supported easy COM communication for control of the Web browser.

The early versions of Arakne supported navigational hypermedia as before, as well as guided tours through Ariadne. The DHM hypermedia server supported multiple users, but had no special provisions for collaboration. As all link decoration was handled through the DHMProxy, the task of the Arakne hypermedia tools were limited to handle authoring and link/guided tour browsing. Communication with the DHMProxy was handled through overloading URLs with instructions to change hyperspace etc. Interface wise, Arakne operated (and still operates) with the MDI (Multiple Document Interface) idiom — where the hypermedia tools ("views") are contained as windows within a larger window. As described above it was the experience that small windows tended to become hidden behind other windows on the desktop, and the move of all (small) hypermedia tools into one larger window eliminated this phenomena. The downside was (and still is) that Arakne was fairly large, which could be annoying on smaller screen. This however is less of a problem that it would have been with a Netscape browser, as the Microsoft Internet Explorer supports the addition of tools in the context (i.e. right click) menu. By adding the most common operations (such as "Create Link", "Add Endpoint", etc.) to this menu, user only need to switch to Arakne for link browsing or other more advanced tasks.

The design of Arakne was based on the Arakne Framework (described in Section 6.1) with an almost one-to-one mapping to classes. One of the aims of this design was to create a modular system, where single components (such as the Browser class responsible for controlling a Web browser) could be replaced without affecting the rest of the system. By implementing event listening interfaces, components (e.g. a view) could subscribe to events, such as browsing events.

Since the initial Arakne prototype, the system has seen two major revisions, both

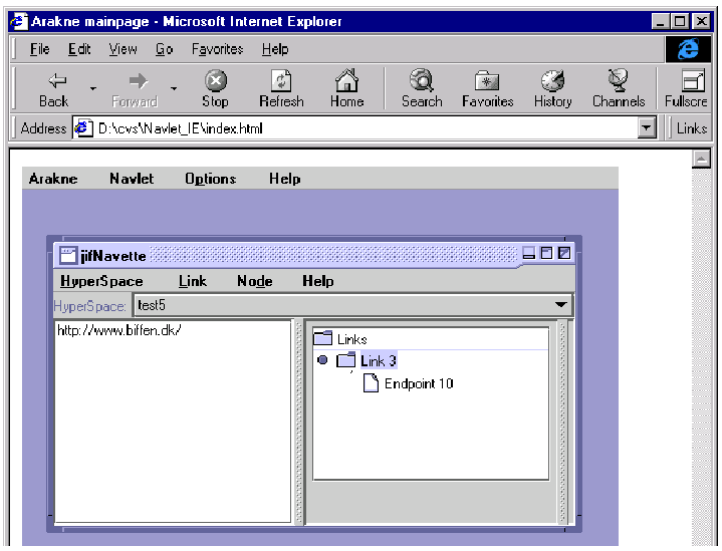


Figure 6.7: Early Arakne Version

involving the move from DHM to OHP hypermedia servers. The experiences with the move to OHP, described in Section 6.3, showed that this was the case to a certain degree, as it was indeed possible to move to Construct without greatly affecting the interfaces to the views. This convenience however happened at some considerable cost, codewise. The component identified as “Operations” in the Arakne Framework (Figure 6.1) was dealing with Construct servers, and thus used the Construct data model and the associated classes. The top layers, that is the views and the interface to the Hyperstructure Store used the data model inherited from the previous version of Arakne. This older data model had some advantages over the Construct model. It was, from a navlet developer’s point of view, less cumbersome to work with, and fitted more directly into the Java idiom. It natively supported the model/view paradigm, so that hypermedia objects alerted their viewers or associated objects of changes made to them. Thus, a developer could create a few event listeners and be alerted to any changes that took place within the data model. This, while very nice when developing views, proved to a debugging and maintenance nightmare, as well as imbuing a considerable overhead to the system. All elements in the Construct data model has to be duplicated in the Arakne data model by wrapper classes that maintained the original functionality. Clearly, there was need for some innovation. It also became clear that there was still room for considerable improvement in the design to facilitate greater flexibility with regards to future expansion. The early Arakne Environments relied on the “Hyperstructure Store”, one class consisting of convenience methods for all navlets. We wanted to maintain a convenience layer, as it eases development, but having this functionality in one class was clearly a problem. This class was ever growing in size and in complexity, and a modular approach was in order.

The new design had to fulfil certain requirements:

- New functionality (such as new structuring mechanisms) should not influence existing code

- Modular and dynamic architecture — no large monolithic classes, and no instances of unneeded classes
- "Plug and play" functionality — updates and new releases of views or services should be automatically integrated into the system
- Conceptually simple for the view programmer
- As much code reuse from Construct as possible — no duplicated efforts

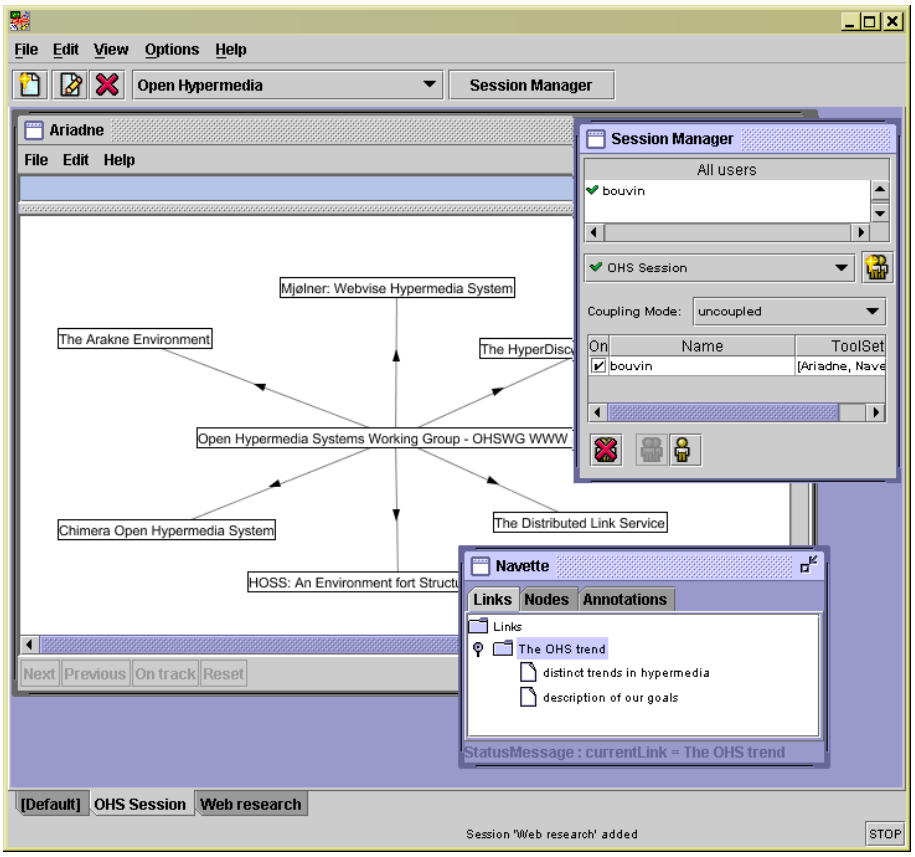


Figure 6.8: The Arakne Environment: Two sessions, two views, and a Session Manager. The ticker tape is visible at the bottom. The tabbed panes just above the ticker tape is used to switch between joined sessions.

The end result of this redesign is the current version of the Arakne Environment, seen in Figure 6.8. When Arakne starts, it checks whether its constituent classes, located in JAR (Java Archive) files, are up to date by checking a distribution Web site. If a new version has been released, the new JAR file is retrieved and made available for the system. Through introspection, the Arakne Environment determines which views and services are available, and updates its user interface accordingly. If a user launches a view, the necessary service (or core) is instantiated and made available for the view.

New views and matching cores as well as central Arakne classes can be added to the distribution Web sites, and seamlessly integrated into the existing system.

This functionality is facilitated by two new central classes: The `JarLoader` and the `DynamicEntity`. The former is responsible for retrieving updates from the distribution Web site. The latter handles the instantiantion of the views and their services. Neither of these classes require any modification for the addition of new services or views. There is still a convenience layer in Arakne, but now it has been delegated to the individual services, and the structure of these convenience classes has been streamlined considerably to the point, where they eventually can (and in all likelihood will) be auto-generated to a large degree.

In some ways the convenience layer is "thinner" than it was before — the view developer is closer to the core functionality. However, this cost is offset by the lower development cost of the services due to a simpler design, which should also reflect in the number of bugs encountered in the code.

Some steps are still missing, before the Arakne Environment can be considered really easy to develop for. One thing is developing new views for existing services, such as a replacement for the navigational hypermedia tool, Navette. In such a case, the existing services are readily available. However, a system should also support the addition of genuinely new functionality, such as a new structuring mechanism (e.g. taxonomic hypermedia). In some aspects, the Arakne Environment supports such extensions very well, as it will automatically update and reconfigure itself for new services and views. In the development phase however, specialised development tools, like CSC [105], would allow more rapid prototyping of new services. Such tools can certainly not take the hard work out of making hypermedia applications, but it can address some of the drudgery.

6.7 Summary

I have in this chapter detailed the work I have done during my Ph.D. During that period I have with DHM/WWW as a starting point developed increasingly sophisticated Web augmentation tools leaving me today with the Arakne Environment, a general architecture for Web augmentation tools based on the OHP compliant Construct servers. The Arakne Environment supports not only a number of hypermedia structuring mechanisms, but also session based collaboration. My latest work, the iScent framework, seeks to develop a scalable architecture to support shared awareness and intersubjectivity. This fits within the Arakne Environment, but can also be used in many other settings, where tools can be integrated to generate events, reporting the user's activities.

Chapter 7

Challenges for Web Augmentation

Throughout the development of prototypes, some lessons about Web augmentation are learned. This chapter sums up the experiences not directly related to Web augmentation per se, but still important. A dominant theme in my work has been the challenge of working with standard technology and existing standards. In open hypermedia we do not have the luxury of writing everything from the bottom, we have to adapt our systems to existing systems. In many ways, the experiences learned with integrating with Web browsers over these past years are prototypical for open hypermedia. Another challenge for Web augmentation is that of scalability. The foremost quality of the Web is its scalable nature. Apart from the added functionality of Web augmentation, it also becomes important that users should not feel hampered performance wise, when using Web augmentation. Likewise, it should be possible to design Web augmentation systems able to withstand the load of many concurrent users. Some advances made in this area through the active use of interchange files are presented.

7.1 Working with Web Browsers

One inescapable consequence of my client-side approach to Web augmentation is the necessity of integrating existing Web browsers. Open hypermedia is based on the tenet that hypermedia must be integrated with the applications, users are already using[1]. An important part of developing Web augmentation hypermedia systems has thus been working with Web browsers. This section describes my experiences with Web browser, going into details on how the Web browsers over time has influenced (or hampered) my work.

The work on Web augmentation at the Department of Computer Science at University of Aarhus started in 1996, at a time where the dominant Web browser was the pre-3.0 Netscape Navigator. Java applets in Web browsers was a fairly new phenomenon, and the new Navigator 3.0 had just introduced the integration between Javascript and Java. This integration made the early browser integrations possible. The DHM/WWW Java applet (Figure 6.5) retrieved Web pages, as well as links and anchors, merged the

[1] Interestingly, the Web is an exception from this. The Web is a monolithic hypermedia system, yet has managed to conquer the rest of the world, regardless of platform and operation system. No other system (and certainly no hypermedia system) has, to the author's knowledge, ever achieved this feat.

twain, and displayed the result in a Netscape window by generated Javascript code that wrote the actual pages. This approach was quite slow, regardless of whether the Web pages in question contained any external anchors. Furthermore, the Netscape browser did not allow selections to be detected, nor the Java applet to access the displayed document in any way — this resulted in a rather cumbersome arrangement, where the Java applet would cache an unmodified copy of the displayed document for reference purposes. Link creation consisted of copying a context for the desired anchor to the Java applet, and selecting the text to be the anchor in this copied context (this two step approach was taken to improve probability of correct anchor identification). The Java applet would then proceed to find the context and the anchor within the cached copy of the original document. It should be noted that due to the so called "sandbox" security constraints imposed on Java applets, the DHM/WWW applet was restricted to retrieving Web pages exclusively from the Web server, wherefrom it originated. While the document displayed in the Netscape browser as such was closed to the Java applet, the converse was not true, as it was possible to call the Java applet through the use of Javascript. Under ordinary circumstances, the act of following a link would render a Java applet impotent to influence the Netscape browser. To avoid this, the DHM/WWW applet replaced all occurrences of ordinary `<a href` links with links that through Javascript called the DHM/WWW applet. Thus, the applet was able to retain control under normal browsing. Of course, as soon as the user entered a URL into the Netscape browser rather than the DHM/WWW applet or used a bookmark, this control was irretrievably lost.

These circumstances all lead to an inefficient, cumbersome, and fragile prototype. The only advantage of the DHM/WWW applet over the systems I have later designed and developed was that of platform independence. DHM/WWW worked on all platforms supported by the Netscape browser.

Basic features, such as selection detection, was not available, and the coming of the next version of the Netscape browser was eagerly awaited, as improvements had been announced. The next version of Netscape, version 4.x, opened up new possibilities for Java applet based Web augmentation. Recognising that the Java "sandbox" was too restrictive, the notion of digitally signed Java applets were introduced[2], allowing trusted Java applets to gain a wider access to the Internet, the local file system, or, especially relevant to our cause, the Netscape browser itself. This version of the Netscape browser supported an event system, where actions such as `mouseOver`, `mouseClick`, `onSelection`, etc. could be captured through Javascript. This finally made it possible to determine selections on a Web page for elegant link creation, and to capture attempted link following without modifying every link on a Web page, as was necessary with DHM/WWW. Unfortunately, even with digital signing, the Netscape browser did not permit capture of `onSelection` events across frame boundaries, and link creation was thus still a two-step process. Through improved, the Netscape browser did not allow post-render modification of Web pages. To accommodate the pre-render modification of pages while trying to avoid the high performance hit of generating Javascript to generate the entire page, a trick, possible only on the Netscape browser was employed. It is through digitally signed Javascript possible to change proxy-settings dynamically. Thus, upon start, Navette (Figure 6.6) would launch a small local proxy, and setup the browser to use this proxy. If the browser already were configured to employ a proxy, this proxy was in turn used by the Navette proxy. While a

[2]It should be noted that at this point, there were three different signing standards for Java applets: Sun, Netscape, and Microsoft. This situation has not improved, and digitally signed Web browser independent Java applets are thus impossible.

marked improvement over the approach taken by DHM/WWW, this approach was still fairly slow, as it (discussed in Section 4.3.9) invariably slowed the retrieval of all pages, regardless of the amount of hypermedia content.

At this time, the Webvise client was being integrated with the Microsoft Internet Explorer. This browser featured a COM interface which not only made it possible to control the browser itself, but also access and modify the Web page currently displayed. This allowed the Webvise client (as the first open hypermedia client to the author's knowledge) to perform post-render link decoration on a standard Web browser. This was much faster, and considerably more elegant, and clearly the way to proceed. Thus, the Netscape browsers, who did not feature such interfaces, were abandoned in lieu of the Microsoft Internet Explorer.

This move had of course some consequences. The Netscape browser supported more computing platforms that the Microsoft Internet Explorer, and these platforms were lost in the move. Furthermore, the integration with the Internet Explorer relied on using a COM interface, which required us to use the proprietary Microsoft version of Java, which explicitly supported this. Thus, even the platform independence of Java were lost in the move. Yet, some of these losses were acceptable. At this, the so called "Browser War" was nearing its end with Microsoft as the victor. By switching to the Internet Explorer, we were supporting the most widely used Web browser. The loss of platform independence was regrettable, but at least we were supporting the most widespread operating system, Windows. It should be noted, that not all platform independence was lost, as only the Web browser integration code relied on the COM interface. Thus, only this platform specific code need rewriting, if we at a later time should target a new Web browser on a different platform.

The first iterations of the Arakne prototype relied on Microsoft Java for COM integration. Unfortunately, as tensions heated between Microsoft and Sun, the former stopped releasing new version of their Java tools. This locked the Arakne prototype into Java version 1.1 for quite a while, and thus making the user interface and other improvements made in Java 2 (version 1.2) unavailable. At this point, the move to Construct was well underway, and as Construct relies on the collections classes found in Java 2, the Construct used in Arakne had to be rewritten to accommodate Java 1.1. This was an unacceptable development overhead, and the decision to move to abandon Microsoft Java was taken. This was possible only because of a commercial Java/DCOM integration, JIntegra, which we started to use. This leaves the Arakne Environment where it is today. The prototypes which started out as platform independent if browser dependent applets have today evolved into an application supporting only one browser and one operating system.

7.1.1 The Ideal Web Browser

What would, from a Web augmentation standpoint, constitute the ideal Web browser? As described in the previous section, most of my Web augmentation work has been hindered as much as helped by the available Web browsers. Especially the development utilising the Netscape browsers were frustrating. Clearly, there must be a better way.

As describing in Section 7.2, I have throughout development used Java. As the first Web augmentation tools were Java applets, this was a natural choice. Today, this development however leaves an application, that cannot reap the platform independence benefits of Java, but must suffer under the Java platform's weaknesses, such as (relatively) slow user interfaces, relatively large memory requirements, and difficulties with integrating native tools.

In many ways, the current Internet Explorer is a good candidate to what a Web browser should provide for prospective Web augmentation developers. It is modular, and can be integrated into other applications, and other applications can be integrated into it (as done by the tools described in Section 5.2). It has a rich interface, both for control of the browser, and for manipulating the Web pages displayed by it. On the flip side, it is predominantly Windows oriented, and requires COM for integration. The newest version of the Internet Explorer (version 5.5) purports to support editing of HTML files. We have only recently begun conducting experiments in this regard, but the prospect is quite interesting.

The ideal future batch of Web browsers would support the functionality of the Internet Explorer, but doing so by relying on a general cross-platform and -browser independent API, accessible regardless of programming language. This would simplify the development of Web augmentation tools, and hence make the development of such tools much more attractive. This could be accomplished either over a process to process communication protocol (such as COM, Corba, or RMI), or by some scripting language (i.e. accessing the internals of the browser through e.g. Javascript).

`http://www.mozilla.org/`

As such, the Mozilla browser holds some promise with its XPCOM component model. It is still too early in the browser's development process to see whether it will be able to compete with Internet Explorer. Given the open source nature of the project, it should be possible to create an API for Web augmentation tools. This however is future work.

7.2 Experiences with Java

All the Web augmentation systems developed prior and during my Ph.D. has been implemented in Java. The first prototype (pre Ph.D.) was DHM/WWW and was implemented in 1996. During the course of these four years, the Java language has matured considerably and this section will reflect on the experiences with Java development.

By far the greatest drawback of using Java has been the problematic support of COM. The Internet Explorer is handled through COM, and thus the ability to handle COM is crucial. Over the last few years, Sun and Microsoft has been engaged in a lengthy legal dispute, with COM support as one of the casualties. Thus, the current incarnation of the Arakne Environment relies on a commercial third-party DCOM integration, JIntegra. This makes distributing Arakne problematic, as the license for JIntegra is quite expensive.

`http://www.linar.com/`

In hindsight, it is fair to question the decision to stay with Java throughout the development process. Certainly, the Internet Explorer integration would be much easier, if the Arakne Environment was written in `C++`. On the other hand, the last two versions of the Arakne Environment has reused much code from the Construct effort — code we would have been forced to reimplemente, had we not used Java. Clearly, there are pros and cons to this question. Apart from the Construct code, we have also benefitted from the extensive standard libraries found in Java. There are things, that could be better in Java (such as the lack of templates), but it is a very accommodating language to develop in.

If I was starting from scratch today, given the current Web browser situation, I think the wisest course had been to use C++. This would have allowed a much closer integration with the Internet Explorer. One problem that remains is that this would effectively lock us with the Internet Explorer. Currently, there are no real challengers to this Web browser, but this may change in the future — once the Netscape browser was com-

pletely dominant. If the Arakne Environment in the future should continue with Web augmentation (it might also extend to other types of applications), the system should not be tied to one browser. If a new browser was to emerge today, and if it provided in one way or another an interface similar to the one found in the Internet Explorer, porting the Arakne Environment to support this browser would require rewrite of a few classes, but it would not affect the rest of the system.

My work has been on the client side of open hypermedia, and I have done relatively limited development on the server side. On the server, the benefits of Java are much clearer. The ability to run servers anywhere (or at least where Java is supported) should not be underestimated. Likewise, the platform independence, most Java programs should exhibit eases code sharing between different system considerably. Currently, both the Construct and the Chimera backends are implemented in Java. Performancewise, the penalty of Java is not so great as on the client. Much of the heavy calculations should in most modern hypermedia servers be relegated to the database backend.

Javasoft has recently announced that they in the next quarters are going to focus on the client-side of Java. The last few versions of Java has shown considerable progress, and it is to be hoped that further progress will be made. The most essential, at least from my point of view, is the ease with which COM communication can take place. Prior to the move to JIntegra, we conducted some experiments with the Javasoft Java/COM Bridge. At that point, it did not meet our requirements, and this is clearly an area, where Javasoft could, to our benefit, focus its resources.

`http://www.javasoft.com/`

7.3 Hypermedia Scalability

One of the classic problems with any hypermedia system relying on externally stored hypermedia structures is that this impacts scalability. Rather than just retrieving and displaying a document, a hypermedia system also has to query its hypermedia servers for relevant structures, retrieve these structures and display them in conjunction with the document. Evidently, this is more computational and time intensive than just retrieving a Web page, and scalability is indeed one of the Web's strong points. Web augmentation will invariably be compared to what can be achieved with ordinary Web technology, and as such scalability becomes a great concern for the Web augmentation field, if it is ever to see widespread use.

In the context of this discussion I would like to approach scalability from two different angles: Scalability in the sense of concurrently serving many users, and scalability in sense of serving massive hypertexts. While both are highly desirable, they put different stresses on the overall system. The imperative of a myriad-user system is low latency, that is completing single operations quickly. This is, where the Web shines.

The problem of supporting massive hypertexts is slightly different, and more complex. Anderson has done a lot of work on achieving this form of scalability with the Chimera system [3]. The hypermedia system must both on the server and the client side be able to handle large structures. On the server, this implies large scale storage, which in practice translates into using a solid database, in the case of Chimera, PostGreSQL. Clients are usually built using standard GUI widgets, and these do often not scale well, when confronted with tens of thousands links etc. One thing is the scalability of GUI widgets, another question is if standards GUI widgets such as lists, trees, and tables remain an effective interface for managing many items. One approach taken by Anderson was to implement filtering, so that users were only presented with the links of imme-

diate interest. Given that the client and the server is able to handle massive hypertexts, the protocol between them must be considered. For instance a protocol, where each link is retrieved one at a time imbues an enormous overhead on the system, in comparison with a protocol where links are retrieved en masse. When replies to queries can be large, it becomes relevant to modify the protocol, so that for instance the size of the reply is reported before the entire reply is returned. This would allow the client to inform the user of the retrieval in progress. Likewise, a client should be able handle replies as they are being received, thus improving the apparent responsiveness of the system. An interesting topic in this area is the visualisation of massive hypertexts. If standard GUI widgets break down both with regards to both performance and usability, new forms of interacting with such large collections must be devised. This is a difficult problem, and one I for the Arakne Environment has relegated to future work.

The Coconut project has also been working with scalability issues, and we have investigated two approaches. Firstly, one could build an infrastructure based on a solid database such as Oracle, and by utilising the industrial strength of these systems achieve solid performance. Secondly, one could try to do away with the hypermedia server altogether. The Construct servers were originally based on a relational database backend, and for large scale usage this still seems to be the proper course, given the possibilities of backup and security that these databases offer. However, the performance of the Construct database never became satisfactory, and it was thus abandoned in lieu of a simpler, but faster, file-based solution. Future work on the Construct servers would certainly benefit from more work and optimisation on a database backend, so that the advanced features of these systems, such as transactions and rollback could be used to provide greater fault tolerance. The developers of the Chimera system [2, 5, 7, 8] at University of Colorado, Boulder, has had good experiences with database backends, so it can be done satisfactorily.

7.3.1 Scalability through Interchange Files

Another approach to attain scalability is to eliminate the bottleneck of the hypermedia server, or (more correctly) to minimise the strain on the hypermedia servers. This can be accomplished through the use of hypermedia interchange files. When an author of hypermedia structures has finished working on a particular set of structures (typically a context), it is saved into an interchange file, which can then be freely distributed. An example of such an interchange format is the Open Hypermedia Interchange Format (OHIF) [55], another is link base files in XLink [37].

The Open Hypermedia Interchange Format is strongly influenced by the data model underlying the Open Hypermedia Protocol. Currently, OHIF supports navigational hypermedia and guided tours, but the format has been designed to easily accommodate future expansions. Compared to XLink, there are some similarities, as both are XML based, and both offer *n*-ary, bi-directional links. Both XLink and OHIF can through locators respectively locspecs address selections in arbitrary document types[3]. While navigational hypermedia still is the mainstay of hypermedia, there are many other structuring mechanisms, that cannot easily (or at all) be modelled by links, such as compositional, spatial, taxonomic, or issue-based hypermedia. As OHIF can be extended to handle such structures, it is, in this aspect, a stronger interchange format than XLink. Currently, OHIF is supported by Webvise and the Arakne Environment.

[3]Currently, XLink seems to be largely dependent on XPointer, which only addresses XML documents. The standard is however open for other specifications.

That interchange files contributes to scalability can be illustrated by the deployment of Ariadne [66] at Opasia, discussed in Section A.2. The approach in this instance was more extreme than the one taken by Webvise and Arakne, as the latter two still employ a hypermedia server to handle the interchange file, whereas Ariadne has no server interaction apart from downloading the interchange file (a retrieval handled by a Web server). It should be noted though, that the Ariadne deployed at Opasia was a read-only version, stripped of the ability to author guided tours. Thus, there is no need for a hypermedia server.

This approach solves only some scalability issues. It certainly allows many users to (read-only) access the same hypertext, and it provides collaborators with an easy way of turn taking on hypertext authoring. It does however not address the larger problems of simultaneous authoring access for many users or massive hypertexts.

7.4 Summary

I have in the preceeding sections detailed some of the obstacles to Web augmentation, that I through my work have encountered. These are not an exhaustive list, but should be fairly representative. Being a subfield within open hypermedia, Web augmentation shares many traits with the larger field, such as third-party application integration. Web augmentation is not very interesting, lest it provides the user with a full hypermedia integration (otherwise the improvement compared to unaugmented Web would be slight), and as such integrations are a top priority. Partly due to the “Browser War”, Web browsers over the last four years have seen tremendous development. The positive aspect of this in conjunction with my work was that if a feature was not present in a Web browser, it probably would be in the next release a quarter later. The negative aspect was that these newly implemented features (especially in the case of Netscape) were often poorly thought out or implemented. With the move to the Microsoft Internet Explorer, some stability has been achieved, but at the cost of tying Webvise and the Arakne Environment to one Web browser and one platform. Interoperability between different platforms is always tricky, as has certainly been the case with Java and Windows. Whether this situation will improve remains to be seen.

One sensible requirement of Web augmentation is that it should work well with the user’s Web browser, another is that it should not degrade the overall Web experience by for instance slowing retrieval of Web pages down. On the client, this has been addressed through post-rendering link decoration. The question of designing Web scalable open hypermedia systems still remains open, though it in the context of Web augmentation partly has been addressed through the use of interchange files stored on Web servers. Interchange files are well suited for publishing hypertexts, or for turn taking co-authoring. Future scenarios of many users actively and simultaneously co-authoring hypertexts on the Web cannot be handled through interchange files, and thus there are good prospects for future work in this area.

Chapter 8

Conclusion

I have in the preceeding chapters described my work on Web augmentation during the course of my Ph.D. I have developed several prototypes, published papers about them and their underlying models, as well as worked on OHSWG standards and future systems for intersubjectivity support. My experience is that the realm of the Web is definitely open to open hypermedia.

This chapter sums up my experiences and outlines my plans for the immediate future, as well as some (hopefully educated) guesses on the long term prospects of Web augmentation as a subfield of open hypermedia and the World Wide Web.

8.1 Primary Results

Throughout my Ph.D., I have designed conceptual models for Web augmentation, and subsequently implemented a number of increasingly ambitious Web augmentation systems. These systems have demonstrated the viability of Web augmentation as a technology. As the Web stands today, there is still ample room for improvements with regards to the structural and collaborative mechanisms available.

The Arakne Environment embodies an approach to Web augmentation that if it does not attempt to do all things Web augmentation, at least accommodates the possibility through its modular design. Especially the integration with Construct with its potential limitless amount of new hypermedia structuring mechanisms [105] holds great promise. The Arakne Environment is not architecturally bound to one platform or Web browser, and should thus prove adaptable to future developments.

Web augmentation allows users to structure their Web experience and to share this with other users of the Web. As a tool it can find use in both professional settings as a knowledge management and structuring tool, as well provide Web surfers or Web journalists with more powerful tools. The last few years have seen a proliferation of meta Web sites, that largely reports on occurrences on other Web sites. Web augmentation fits nicely within that niche.

The Web is not rendered obsolete by Web augmentation, quite the contrary. Web augmentation strengthens the Web by extending the possibilities of interacting and using the Web. Ultimately, Web augmentation is a question of liberating the link, and allowing users to participate in a free, Web-wide discourse by linking, associating, annotating, and restructuring the existing Web to fit their need and vision. An important task for the future is to devise an architecture able to withstand such large scale use.

Some steps has been taken in this direction with the work on the Open Hypermedia Interchange Format.

I doubt that the Web will on a large scale be succeed by new technology anytime soon. The Web is easily "good enough" to satisfy current and many future needs without warranting replacement. The Internet, propelled by the Web, was a revolutionary change in the general availability and use of computers as tools of communication. When the Web emerged, or rather erupted, it filled a previously unfilled niche. Now that the Web reigns supreme, future changes will in all likelihood be gradual rather than sudden. Web augmentation is one such gradual change, improving the Web without replacing it. Future changes to the Web will take place within a Web context, not without it. This is also witnessed by the growing proliferation of Web technologies in other contexts, such as the spread of XML. The Web and the Internet are increasingly becoming infrastructures — the substrate upon which computing is built. One technology that *may* be challenged in the years to come is the client/server model underpinning the Web in the sense that the distinction between clients and servers will become blurred. There will be many more behemoth servers serving increasingly sophisticated content, but simultaneously the common household computer will be connected permanently to the Internet. The permanence of connections coupled with the generous storage and processing capabilities of modern computers enables a more widespread use of peer-to-peer networking. This has already been convincingly demonstrated by file sharing tools such Napster or Gnutella. While these tools predominantly has been used for software and content piracy, the very model of clients connecting to clients is viable. An interesting project is this context is the Freenet Project [28], which seeks to create a global docuverse, with all content encrypted to the degree that a maintainer of a Freenet node cannot determine what is stored on that node.

`http://www.napster.com/`

`http://www.freenetproject.org/`

Web augmentation as a technology is open hypermedia, which in its eleven year old history has demonstrated its wide applicability. Web augmentation as such is limited to the Web, but as has been demonstrated by e.g. Webvise [56] and Mimicry [20] [P2], Web augmentation does not end with HTML. The unique advantage of Web augmentation as a subfield of open hypermedia, is that the Web in general works with open formats and standards. While proprietary differences in implementations exist and may hamper work, the principle of openness still holds. Compared to the general work of open hypermedia dealing with applications with often closely held proprietary document formats, the subfield of Web augmentation has it much easier. The possibilities inherent in the Web architecture (such as creating specialised hypermedia proxies), or the combination of dynamic HTML with scripting and stylesheets are many indeed, and I am certain we have only begun to scratch the surface of what can be done.

8.2 Web Augmentation in the Future?

A work such as mine, aimed as it is to better the current state of affairs, should also be considered in the longer term. Apart from technical concerns, such as the availability of a certain Web browser or or a specific programming language, the long term viability of Web augmentation as such must be considered.

The most probable successor to HTML is the XML based XHTML. *If* the Web evolves into a space of XML documents[1], initiatives such as XLink (see Section 5.1) can come into play. In such a context, where XLink has been adopted by the leading

[1] This is not given — the inertia of the existing approximately 1×10^{12} Web pages, much of which does not conform to any known HTML standard whatsoever, is considerable.

Web browsers, some of what the Arakne Environment can accomplish today will be directly supported in the Web browser. In such a best-case scenario, the Web would have evolved into a "proper" hypermedia system [82], allowing arbitrary linking and annotations to take place. Furthermore, a XML future could also entail that other document types, such as word processing files or spreadsheets, would migrate to XML, and would then also be subject for easy linking. What would be the role of the Arakne Environment or indeed any Web augmentation tool in such a context?

My immediate reaction to such a scenario is very positive. The purpose of my work has been to investigate the possibilities of Web augmentation, not to develop *the* Web augmentation system of the future. I wish Web augmentation and the possibilities it gives people to become widespread, so any technology that can bring that about is fine by me.

While base functionality such as authoring of navigational hypermedia would reside in the Web browser, the hypermedia research community has developed so many other exciting hypermedia structuring mechanisms, that there would still be ample room for Web augmentation research. While the basic ability to author links is an essential part of most hypermedia systems, another problem with the Web also warrants our interest, namely that "getting lost in hyperspace" with a corpus as large and as unstructured as the Web is a common phenomena. Some technologies, such as spatial hypermedia [74, 89] or advanced structure visualisation, hold some promise to help us navigate this jungle of information. In this regard, the community driven meta Web sites mentioned in Section 4.7.3 are interesting. They demonstrate the viability of thousands of Web surfers scourging the Web for interesting stories and reporting the cream of the crop in a common forum. If this could be moved into a large, ever evolving hypertext with many authors and moderators, such a system might be the tool to handle the complexities of the Web.

Another possible future is that of the heterogeneous Web — while there will be a move to XML, this will not be nearly complete, and thus essentially leaving us with a Web not too different with what we have today. Additionally, the future Web will most likely be accessed not only from more or less comparable (and increasingly powerful) personal computers, but also from a growing number of appliances[2], such as PDAs, mobile phones, etc. Such devices have fairly limited displays, and the work to provide good hypermedia functionality on such devices will be interesting.

8.2.1 Commercial future of Web augmentation

As described in Section 5.2, there are several commercial ventures into Web augmentation. Whether these will survive, or suffer the fate of many dot coms remains to be seen. My earliest fascination, when introduced to the hypermedia research litterature, was that of the trail blazer, the professional creator of associations. The question is: Can you sell links on the Web? The described companies seem to focus on making partnerships with other companies, e.g. essentially advertising. One business model that might work is the combination of an informative Web site, which in addition to its general services could offer its subscribers an "enriched Web" through Web augmentation. This was the vision at Landbrugets Rådgivningscenter (described in more detail in Appendix A.1). Another possibility would be use Web augmentation to attract more users to a Web site, as done by Opasia with Ariadne (see Appendix A.2). The current

[2] According to several keynotes at WWW9, the number of such devices connected to the Internet will surpass that of personal computers in a few years.

Web is largely financed on an advertisement basis, and whether this model holds for Web augmentation remains to be seen. The advert of micro payment (to the author's knowledge still a technology under development) may very well change that, though it of course is a question whether it will meet with general approval from the Web's users.

8.3 Future Work

In many ways my Ph.D. has been mainly establishing a base for future work. The real applicability of Web augmentation can only be tested in a setting with mature tools, and only recently has the Arakne Environment reached a sufficient level of usability. As such there still remains much to be done.

8.3.1 Collaboration in the Arakne Environment

The Arakne Environment provides the collaboration support found in the OHP compliant Construct servers. This functionality of one of the newest additions to the system, and quite likely also still a source for bugs. Collaboration support is notoriously difficult, and it will not suffice that Arakne Environment performs as expected, if this is not the functionality needed in a given work setting. The Arakne Environment has never seen use in other organisations, and until such testing has taken place, it is too early to evaluate the Arakne Environment as a collaborative environment. While prototypes and mock-ups certainly has their place in the design process, I believe that applications can only *really* be tested, when they offer full functionality. One of the hard lessons of my Ph.D. is the time it takes to get to that point.

I am quite excited about the iScent framework. As planned, the system offers to be an externalised memory for the individual user as well the projects, the user participates in. One part is the collection of events documenting a user's actions, another (and much more interesting in my opinion) is the user's retrieval, analysis, and use of these events. The client to handle this, the Trail Viewer, is quite prototypical at the moment, but we (Ken Anderson and I) hope to develop it into an application supporting advanced structuring of events. This will (of course) take place within the Arakne Environment. One area, where I plan to extend the iScent framework is the protection of privacy. The current design suggests that users decide which kind of events they wish to be published. While this certainly protects them from "prying eyes", it also limits their own use of the iScent tools, as they themselves of course cannot retrieve non-published events. An alternative solution would be to introduce encryption of all events at the user level, presumably with some events being private and thus encrypted with the personal key, and with others encrypted with project keys. Thus, only the keeper of the matching keys would be able to retrieve the encrypted events. In such a scenario, users could then, if they so wished, establish "communities of trust" within which the users would be able to see each others' events.

8.3.2 Open Sourcing the Arakne Environment

It is the aim of the author, that the Arakne Environment along with Construct should be open sourced, so that it may benefit more. This decision is reached for the following reasons: Firstly, it seems only proper that computer scientists along with their published results should publish the source code with which they created their results (lest

we become a science of irreproducible results). Secondly, the emphasis especially in the later versions of the Arakne Environment on dynamically extendible hypermedia services and nice coding environments for hypermedia developers would be an exercise in futility, unless it was given a test in the real world.

The main obstacle to open sourcing Arakne is the current reliance on JIntegra for COM integration. JIntegra is a commercial product, and Arakne must thus in the future handle its own COM integration, before it makes any sense as an open source project. This autumn will hopefully bring that about, as well as maturing the code base sufficiently for general release.

Appendix A

Use Studies

Web augmentation has potentially quite wide appeal. In this appendix, I focus on two use studies — one that I have done myself, and another done by other members of the Coconut project. The first case, Landbrugets Rådgivningscenter, illustrates how knowledge workers may use Web augmentation to supply their customers (agricultural consultants) with better services and more in-depth information. The second case, Opasia, shows that Web augmentation can also have wide appeal, in this case Web surfers following guided tours written by Web journalists.

A.1 Landbrugets Rådgivningscenter

Landbrugets Rådgivningscenter is the central Danish agricultural advisory centre. The organisation is controlled jointly by the Danish Farmers' Unions and the Danish Family Farmers' Association. Landbrugets Rådgivningscenter provides service and expertise for the local advisory centres' advisors, who in turn advise the farmers. There are 85 local centres throughout the country. Landbrugets Rådgivningscenter does not generally work directly with farmers. The organisations and their relationship can be seen in Figure A.1.

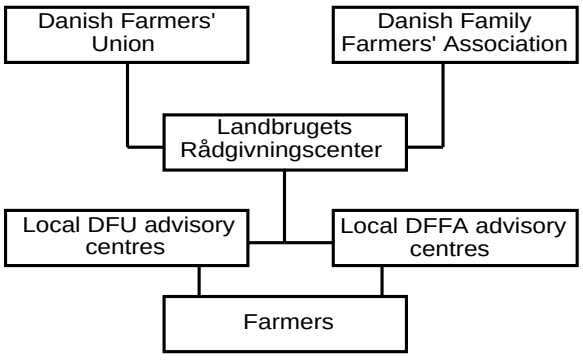


Figure A.1: The Organisation of Danish Agricultural Advisory Services

Landbrugets Rådgivningscenter is organised into the following departments

- The National Department of Plant Production
- The National Department of Cattle Husbandry
- The National Department of Pig Production
- The National Department of Horse Breeding
- The National Department of Farm Buildings and Machinery
- The National Department of Education
- The National Department of Farm Accounting and Management of the Danish Farmers' Union
- The National Department of Farm Accounting and Management of the Danish Family Farmers' Association
- The National Department of Poultry Production
- The Department of Agricultural Law
- International Department
- Section of Ecology

With a very few exceptions (such as pig production, which is predominantly handled by Danske Slagterier (Danish Bacon & Meat Council) Landsudvalget for Svin (Danish Meat Research Institute)), these departments are main source of information related to agriculture in Denmark. The activities include

- Special advice
- Distribution of know-how
- Development
- Experiments and studies
- Education and in-service training
- Maintenance and service tasks

In our work with Landbrugets Rådgivningscenter we have concentrated on the two first areas, this will therefore be described in more detail. The essential task of Landbrugets Rådgivningscenter is to keep the local advisors updated and to be "the experts' experts". The advisors at Landbrugets Rådgivningscenter keep themselves updated with regards to the latest literature and law development and are thus able to answer questions from the local advisors, who may not the time available for this. Furthermore the advisors at Landbrugets Rådgivningscenter keep a close contact to the local advisors, keeping them aware of the needs of their customers.

A.1.1 The Information Series

The distribution of information from Landbrugets Rådgivningscenter to the local centres is primarily done through the information series - publications sent out on a regular basis from the departments. Every department will typically produce several information series catering to specific needs in their field. The information series are copied and distributed by mail on a massive scale (approximately 15 mill. copies/year).

Each publication will typically consist of several articles written by various advisors in the department. Every article is given a unique number to ease archival and reference. The indexing scheme varies from department to department, there is no general pattern. Apart from the individual articles, a few pages of abstracts, briefly describing every article usually accompany every publication. These abstracts are just intended to give the local advisors an immediate overview over the publication, but are often archived at the local centres.

A.1.2 Assembling a publication

Our main case study was at the department of plant production and the description of the process of assembling a publication will therefore be based on this department. We had the process described during meetings with representative from both the plant production and the agricultural law departments. The representatives indicated that the process throughout Landbrugets Rådgivningscenter is comparable to that of the department of plant production.

The advisors at the department of plant production have well-defined fields of expertise, and according to what is happening in their individual field will write articles for publication. Occasionally their supervisor will request an advisor to write a topical article or decide that two advisors who have been writing on related material should work together. The articles will often be inspired by journal articles, new laws, or news, and will as such usually contain quotes from, references to, or abstracts of material not written at Landbrugets Rådgivningscenter. The supervisor maintains an understanding of what the individual advisor is working on — a blackboard outside his office supports this, as advisors write the title of their current work in progress (this also ensures that double work is not done). It is the task of the supervisor and his secretary to ensure that each article is given a unique number. Whenever an advisor finishes an article, he sends the WordPerfect file to the supervisor, who reviews it. If the article is accepted, it is added to the current publication in progress. At the department of plant production, the deadline is early Wednesday. Given this week's worth of articles, the supervisor and the secretary compile the abstracts, check the numbering of the articles and ship the publication to the printing department.

A.1.3 Handling law material

Our initial contact with Landbrugets Rådgivningscenter was with the department of agricultural law. They are mainly occupied with laws and revisions of laws. All Danish laws are accessible on the Web , and the department was investigating, how the Web should affect their work, and how to improve their service to the local advisory centres. While laws are never changed (and thus are a nice stable environment for electronic maintenance) they are often revised. Keeping track of these changes is no trivial task, and it can be quite complicated (at least for a lay person) to establish what is current law, as this requires reading through many revisions. It was therefore one of our early

http://www.retsinfo.dk/

common goals that this process of consolidating the law texts into one document should be supported by our tools. Later the department of agricultural law decided (wisely in our opinion) that maintaining a law Web site was better handled by outsourcing, rather than using their own advisors to keep track of changes.

A.1.4 Moving from paper to the Web

Landbrugets Rådgivningscenter is currently in a transitional phase, as more and more material is made available on the Web. It is the ultimate goal to make the Web the main publishing method. Currently it is the task of the supervisor and his secretary to create the HTML documents. All word-processing at Landbrugets Rådgivningscenter is done with Corel WordPerfect 8.0, and while special macros have been written to ease the translation, it is still no trivial task. The job is complicated by the typographical limitations of HTML, especially with regards to tables. One solution would be to create only PDF-files, which would solve all typographical issues, but PDF is not as searchable or accessible as HTML, so for now the supervisor keeps fighting with HTML. The work of preparing the articles for the Web will eventually move down from the supervisor to the individual advisors, but for now the work is centralised.

Landbrugets Rådgivningscenter is not the only one moving onto the Web. Many of state departments, and private companies and organisations that Landbrugets Rådgivningscenter regularly quote or base their work on, are moving their material onto the WWW. Most noticeable is Pl@nteinfo , which provides the farmers and local advisors with exhaustive information during the growth season.

http://www.planteinfo.dk/

A.1.5 Work with Landbrugets Rådgivningscenter

Meetings between Landbrugets Rådgivningscenter and the Coconut project started in November 1997. The first meetings were exploratory in nature, where both parties were trying to establish whether collaboration would be mutual beneficial. Later meetings focused on the departments of agricultural law and of plant production and their needs with document handling on the Web. These meetings were also used to showcase various forms for Coconut technology. Eventually a collaborative effort between the department of plant production and Coconut project member resulted in a first Web mock-up of just how the future Landbrugets Rådgivningscenter Web site might appear. The first mock-up was a pure technological showcase but was used as input in the development of the second mock-up. This time the basis of the mock-up was a week's worth of articles from the department of plant production and the possibilities that should be explored were decided on during two meetings. Based on this mock-up it was decided that there were good reasons to continue the collaboration. Over the next months, the systems that were to be tested at Landbrugets Rådgivningscenter were further developed.

The meetings with Landbrugets Rådgivningscenter resulted in a wish-list of features and functionality that would be desirable in a future setting. The advisors should in general be able to create links to and from whatever resource they may come across on the Web. These links come in various types, all which should not require the author of the links to have write permission over the relevant Web pages. Based on their input, we created the following scenarios.

> The anchor texts that John D. added were mainly to help readers navigate, and John D. realises that a short comment on the entire law would

be nice. Using Webvise he composes a short paragraph and inserts the comment in the beginning of the page, so that readers who do not read his article still can benefit from his summary. The summary is marked by a comment icon in the beginning of the page (clicking on the icon will expand the comment), so that readers are not required to read John D.'s summary.

The topic of §25 is the amount of manure allowed per area adjusted by soil type and amount of rain. While the soil type is constant for a particular part of the country, the amount of rain is not, and John D. would like to add "live" data on the amount of rain to his article. John D. turns to the Web site of the Danish Meteorological Institute, who has a nice, daily updated table of the accumulated downpour in the regions. The rest of the DMI page is not relevant to the issue at hand, and John D. decides to transclude the table using Webvise. The procedure of create the transclusion is similar to linking, and the effect is that the (always current) DMI table appears a part of the article.

`http://www.dmi.dk/`

Some of terms used in the manual are fairly unusual and to ensure that all advisors out in the country understand, John D. decides to add these terms to the "dictionary". As the other advisors at the department, John D. has permission to modify the shared dictionary of Landbrugets Rådgivningscenter. John D. creates new entries for the terms by creating a new Web page for each term, as this is company policy (an alternative would be to have one large file, but it was felt that this would be cumbersome, as it eventually would be quite large). Having placed his entries at the appropriate place on the Web site, John D. creates a global link from the first occurrences of the terms he encounters in the manual to his definitions. Henceforth any of the terms (in all old, new or future documents) can used to look up the definition.

`http://www.lr.dk/`

Based on these scenarios, we have so far identified the following needs:

- Anchor-based *n*-ary bi-directional links
- Creation of notes and comments to parts of documents. It should be possible to hide these comments from view.
- Transclusions: displaying data from one document in another, so that if the data is modified in the original document, the change is reflected in the other document.
- Creation of links with only a string rather than a document, a position, and a string as starting point. This can be used to create online dictionaries, e.g. any occurrence of the word "camomile" would be the starting point of a link that would end in a Web-page with pictures and descriptions of the plant.
- The ability to create links from and into PDF documents. While Web links can be predefined in a PDF document, no technique exists today to create these links dynamically[1].
- Frame sets are notorious for making linking to a specific set of frames difficult: either you link to the starting frame set or you link to the specific page (and then

[1] According to the Adobe Acrobat Reader license agreement such a tool would violate the agreement.

loose the context of the other frames). Both solutions are unsatisfactory, and a better approach is clearly needed.

- Rather than linking into a piece of text, it should be possible to create an anchor that would appear before or after the relevant section. This anchor would then have anchor text defined at the time of creation, which would be displayed along with the link, when the Web page was displayed.
- The presentation of the external structures above should be very customisable.
- There should different sets of contexts to accommodate different needs (e.g. a plant advisor might be interested in less detail than a law advisor, when reading a law text)
- Exception handling: the change, move or removal of a Web page should be handled gracefully by the system.
- All these features should be present in a reader's only solution, so that advisors and farmers could have the advantages of these sophisticated structures without actually installing any special software.

A.1.6 Experiments at Landbrugets Rådgivningscenter

In late autumn 1998, user testing began at Landbrugets Rådgivningscenter. One user, the supervisor of one department, was elected to be the first adopter, and I was involved in the preliminary testing of Webvise and DHMProxy at Landbrugets Rådgivningscenter.

In February 1999, this lead to a more comprehensive study, where a weeks worth of Web publications were done using Webvise and published through DHMProxy. Selected customers were then to evaluate the system and report back on their experiences. Unfortunately, noone reported back. At the same time, both parties (Landbrugets Rådgivningscenter and the Coconut project) became very occupied with other work, and the experiments ended at this time. This leaves the question, why the customers did not report back. While pure conjecture on my part, I would suspect two reasons: That changing the proxy configuration was too much hassle, and that they did not have time. Clearly, this calls for another attempt.

A.2 Guided Tours at Opasia

Tele-Danmark Internet was the commercial partner in the Coconut project. One of the direct results of the Coconut project was the deployment of a read-only version of the Ariadne guided tour tool on the Tele-Danmark Internet portal, Opasia . I have not been involved with this deployment, but have gracefully been given access the logs of some of the early deployments of Ariadne.

`http://www.opasia.dk/linkguide/guidedeture/`

The guided tours were deployed in December 1999, and by early January 2000, more than 15000 users had browsed guided tours on the Opasia Web site. A link to these tours is featured on the Web site's frontpage , and the service reportedly still is widely used. An example of a guided tour, in this case on the topic of the Olympic Games, can be seen in Figure A.2.

`http://www.opasia.dk/`

The guided tours are written by Opasia's Web journalists using a stand-alone version of Ariadne. Upon completion, the tour is saved in a XML file, and placed on the

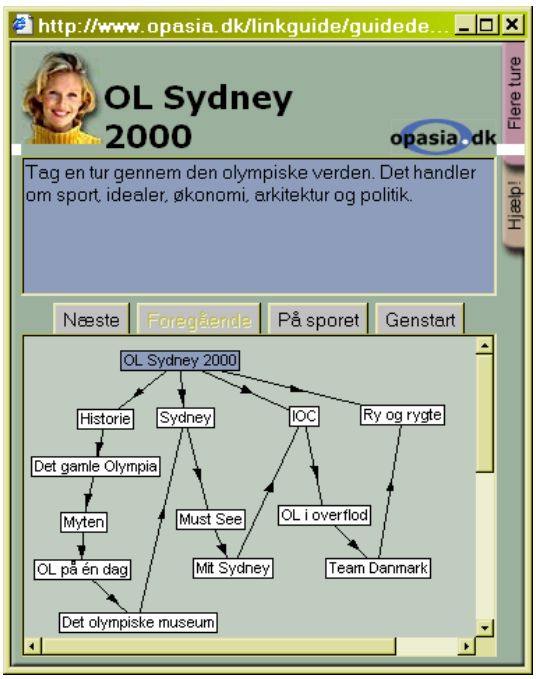


Figure A.2: Guided Tour at Opasia

Web portal. Users can then follow a link, which opens a read-only Ariadne with the guided tour in a separate windows. Topics for guided tours have varied from genealogy to the Euro referendum. There are currently (October 14th 2000) nine guided tours available.

The greatest difficulty encountered in the development of this tool was to slim down the original Ariadne to a small applet, that could run as many different Web browsers as possible. Users may for instance not have the latest version of Java installed, and it therefore necessary to develop for the lowest common denominator.

Appendix B

Installing the Arakne Environment

An important result of my Ph.D. (and a recurring theme in this text) is the Arakne Environment. It can be retrieved from the author's Web site. The text below can also be found at this Web site, complete with links to the necessary components.

`http://www.bouvin.net/Arakne/`

B.1 Systems Requirements

The Arakne Environment currently supports only Windows NT 4.0 and Windows 2000, as it relies on DCOM for the Internet Explorer integration, and this is only found in these operating systems. The system has been tested with the Internet Explorer 4.0–5.5.

The integration with the Internet Explorer is handled through JIntegra, which must be installed. Additionally, the Java Runtime System (version 1.2 or 1.3) must also be installed.

B.2 Installing JIntegra

JIntegra can be retrieved as an evaluation version from Linar Ltd. . When `jintegra.zip` has been downloaded, proceed as follows:

`http://www.linar.com/`

1. Unzip the `jintegra.zip` file into a directory (e.g. `c:\Program Files\JIntegra`).
2. Add JIntegra to the `PATH` environment variable (in this example, add `c:\Program Files\JIntegra\bin`).

B.3 Installing the Arakne Environment

Assuming that JIntegra and the Java Runtime System has been installed, the Arakne Environment can be installed using the Arakne Installer. This program is available at the Web site mentioned above:

1. Download the `install.jar` to a directory (e.g. c:\temp)

2. In a DOS shell, move to this directory, and issue the command `java -jar install.jar`
3. Follow the instructions in the Installer
4. Copy the file `jintegra.jar` found in `c:\Program Files\JIntegra\LIB` to `c:\arakne\Jars\`

B.4 Starting the Construct Servers

In the `c:\arakne` directory, there are three batch files `startNameLoc.bat`, `startMSIS.bat`, and `startHsHm.bat`. Start these batch files in this order, and wait for the servers to be ready.

B.5 Starting the Arakne Environment

The Arakne Environment can then be started using the `runme.bat` file found in the `c:\arakne` directory. It will upon launch check whether its JAR files are current, and if not, automatically retrieve and install them.

Appendix C

Vocabulary

API Application Programming Interface. A set of interfaces allowing programs to interact with a system.

Arakne (English: Arachne) Mythological Greek princess turned into the first spider by the vengeful Pallas Athena.

CGI Common Gateway Interface. A standard for communicating with programs running on a Web server through a Web browser.

Construct The OHSWG compliant server family developed at University of Aarhus and Aalborg University Esbjerg.

Context A set of hypermedia structure entities, such as links, anchors, endpoints, composites, etc. A context is in the OHSWG standard defined as a set of GUIDs. A context can contain other contexts, ´ But not directly or indirectly itself. Known in other hypermedia systems as hyperspace (DHM, Hypervise), hyperWeb (Chimera), link base (DLS, XLink), Web (Intermedia) and so on.

Editor A program able to display and modify some media type, e.g. text, video, Word documents, Web pages, etc.

GUID Globally Unique Identifier.

Hypermedia The combination of contents (e.g. a set of documents of any media type) and structure (e.g. a set of links) that establish relationships between parts of the content. The traditional hypermedia model has been navigational, i.e. links and anchors, but hypermedia is not no longer limited to that kind of structuring mechanism.

Intersubjectivity Reflective shared awareness. I know that you know that I know.

IPC Interprocess Communication.

Java Sandbox Security restrictions put upon Java applets. Java applets cannot contact foreign (i.e. different from their originating Web server) servers, foreign Java applets, other processes, or access the local file system.

Link A relation between endpoints. The classic hypermedia link consisted of two endpoints, a beginning and an end, but this was with Dexter [62] generalised into a set of endpoints.

Link decoration Inserting links, annotations, etc. into a Web page.

LocSpec Location Specifier. A value that specifies a selection in a document of some media type. A LocSpec may contain redundancy to support link recovery. Introduced by Grønbæk and Trigg in [57].

Monolithic Hypermedia Hypermedia systems that rely on specially developed editors, rather than integrating third-party editors. Often, but not always, reliant on proprietary file formats. Examples: KMS, Intermedia, the World Wide Web. Structures such as links may be stored externally of documents.

Navigational Hypermedia Hypermedia structuring with links, anchors, and endpoints. Traditionally, this has denoted what hypermedia is, and characterises many traditional systems, such as KMS or the World Wide Web. Hypermedia today also covers other structuring mechanisms, such as guided tours, spatial or taxonomic hypermedia.

Navlet See View.

OHP Open Hypermedia Protocol. The hypermedia architecture designed by the OHSWG. Supported by the Construct servers and the Arakne Environment.

OHSWG Open Hypermedia Systems Working Group. A group composed of the currently active open hypermedia research groups. The main purpose of OHSWG is to design a shared open hypermedia standard (OHP) allowing all open hypermedia systems to interoperate.

Open Hypermedia Hypermedia systems, that exclusively or largely integrate third-party editors. This generally implies externally stored hypermedia structures, as "foreign" file formats often not will have provisions for linking etc. Examples: Microcosm, HyperDisco, Webvise, the Arakne Environment.

View A tool running in the Arakne Environment (previously known as "Navlets"). Views can for instance be hypermedia tools such as Ariadne, CAOS, and Navette.

Viewer A read-only editor — most Web browsers and image viewers are good examples of viewers.

Bibliography

[1] R. M. Akscyn, D. L. McCracken, and E. A. Yoder. KMS: A distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–835, July 1988.

[2] K. M. Anderson. Integrating open hypermedia systems with the World Wide Web. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8^{th} ACM Hypertext Conference*, pages 157–166, Southampton, UK, Apr. 1997.

[3] K. M. Anderson. Issues of data scalability in open hypermedia systems. *The New Review of Hypermedia and Multimedia*, 5:151–178, 1999.

[4] K. M. Anderson. Scalability in open hypermedia systems. In *Proceedings of the 10^{th} ACM Hypertext Conference*, pages 27–36, Darmstadt, Germany, Feb. 1999.

[5] K. M. Anderson. Supporting industrial hyperwebs: Lessons in scalability. In *Proceedings of the 21^{st} International Conference on Software Engineering*, pages 573–582, Los Angeles, CA, USA, May 1999.

[6] K. M. Anderson and N. O. Bouvin. Enabling project awareness and intersubjectivity via hypermedia-enabled event trails. Submitted for publication.

[7] K. M. Anderson, R. N. Taylor, , and E. J. Whitehead, Jr. Chimera: Hypermedia for heterogeneous software development environments. *ACM Transactions on Information Systems*, 18(3), July 2000.

[8] K. M. Anderson, R. N. Taylor, and E. J. Whitehead Jr. Chimera: Hypertext for heterogeneous software environments. In *Proceedings of the European Conference on Hypermedia Technology (ECHT 1994)*, pages 97–107, Edinburgh, Scotland, Sept. 1994.

[9] K. Andrews, F. Kappe, and H. Maurer. Serving information to the Web with Hyper-G. *Computer Networks and ISDN Systems*, 27(6), 1995.

[10] L. Bannon and K. Schmidt. CSCW: Four characters in search of a context. In J. M. Bowers and S. D. Benford, editors, *Studies in Computer Supported Cooperative Work*. Elsevier Science Publishers B. V. (North-Holland), 1991.

[11] L. Bannon and K. Schmidt. Taking CSCW seriously. *CSCW Journal*, 1(1–2), 1992.

[12] R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkel, J. Trevor, and G. Wötzel. Basic support for co-operative work on the World Wide Web. *International Journal of Human Computer Studies*, 1997.

[13] T. Berners-Lee. The World Wide Web — past, present and future. *Journal of Digital information*, 1(1), 1997. `http://jodi.ecs.soton.ac.uk/Articles/v01/i01/BernersLee/`.

[14] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollerman. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 1(2), 1992.

[15] N. O. Bouvin. Collaborative Web-based open hypermedia and mutual awareness. Submitted for publication.

[16] N. O. Bouvin. Designing open hypermedia applets: Experiences and prospects. In *Proceedings of the 9^{th} ACM Hypertext Conference*, pages 281–282, Pittsburgh, USA, 1998.

[17] N. O. Bouvin. Unifying strategies for Web augmentation. In *Proceedings of the 10^{th} ACM Hypertext Conference*, pages 91–100, Darmstadt, Germany, Feb. 1999.

[18] N. O. Bouvin. Designing user interfaces for collaborative Web-based open hypermedia. In *Proceedings of the 11^{th} ACM Hypertext Conference*, pages 230–231, San Antonio, USA, May 2000.

[19] N. O. Bouvin. Experiences with OHP and issues for the future. In *Proceedings of the Open Hypermedia Systems Workshop 6.0 Hypertext 2000*, number 1903 in Lecture Notes in Computer Science. Springer, 2000.

[20] N. O. Bouvin and R. Schade. Integrating temporal media and open hypermedia on the World Wide Web. *Computer Networks — The International Journal of Computer and Telecommunications Networking*, (31):1453–1465, 1999.

[21] E. Bradner, W. A. Kellogg, and T. Erickson. The adoption and use of 'BABBLE: A field study of chat in the workplace. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 139–158, Copenhagen, Denmark, Sept. 1999. Kluwer Academic Publishers.

[22] M. Büscher, P. Mogensen, D. Shapiro, and I. Wagner. The Manufaktur: Supporting work practice in (landscape) architecture. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 21–40, Copenhagen, Denmark, Sept. 1999. Kluwer Academic Publishers.

[23] V. Bush. As we may think. *The Atlantic Monthly*, pages 101–108, July 1945.

[24] L. A. Carr, W. Hall, and S. Hitchcock. Link services or link agents? In *Proceedings of the 9^{th} ACM Hypertext Conference*, pages 113–122, Pittsburgh, USA, 1998.

[25] L. A. Carr, D. D. Roure, W. Hall, and G. Hill. The distributed link service: A tool for publishers, authors and readers. In *Proceedings of the 4th International World Wide Web Conference*, Boston, USA, 1995. W3C.

[26] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an Internet-scale event notification service. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing*, July 2000.

[27] J. Clark and S. DeRose (editors). XML Path language (XPath) version 1.0. W3C Recommendation 16 November 1999, W3C, Nov. 1999. `http://www.w3.org/TR/xpath`.

[28] I. Clarke. A distributed decentralised information storage and retrieval system. Technical report, Division of Informatics, University of Edinburgh, 1999. `http://www.freenetproject.org/freenet.pdf`.

[29] A. Clement. Considering privacy in the development of multi-media communications. *CSCW Journal*, 2(1–2), 1994.

[30] R. Daniel, S. DeRose, and E. Maler (editors). XML Pointer Language (XPointer). W3C Working Draft 6 December 1999, W3C, Dec. 1999. `http://www.w3.org/TR/xptr`.

[31] C. X. D'Arlach and J. J. Leggett. HyperEd: a spatial hypertext editor. Technical Report TAMU-HRL-94-005, Department of Computer Science, Texas A&M University, 1994.

[32] H. C. Davis. Referential integrity of links in open hypermedia systems. In *Proceedings of the 9th ACM Hypertext Conference*, pages 207–216, Pittsburgh, USA, 1998.

[33] H. C. Davis, W. Hall, I. Heath, G. Hill, and R. Wilkins. Towards an integrated information environment with open hypermedia systems. In D. Lucarella, J. Nanard, M. Nanard, and P. Paolini, editors, *Proceedings of the ACM Hypertext 1992 Conference*, pages 181–190, Milan, Italy, Nov. 1992.

[34] H. C. Davis, S. Knight, and W. Hall. Light hypermedia link services: A study of third party integration. In *Proceedings of the European Conference on Hypermedia Technology (ECHT 1994)*, pages 41–50, Edinburgh, UK, Sept. 1994.

[35] H. C. Davis, D. E. Millard, S. Reich, N. O. Bouvin, K. Grønbæk, K. M. Anderson, U. K. Wiil, P. J. Nürnberg, and L. Sloth. Interoperability between hypermedia systems: The standardisation work of the OHSWG. In *Proceedings of the 10th ACM Hypertext Conference*, pages 201–202, Darmstadt, Germany, 1999.

[36] Delta-V Working Group. `http://www.webdav.org/wg/#dv`.

[37] S. DeRose, E. Maler, D. Orchard, and B. Trafford (editors). XML Linking Language (XLink). W3C Working Draft 21 February 2000, W3C, Feb. 2000. `http://www.w3.org/TR/xlink/`.

[38] P. Dourish and S. Bly. Portholes: Supporting awareness in a distributed work group. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 541–547, May 1992.

[39] D. Engelbart. A conceptual framework for the augmentation of man's intellect. In P. Howerton, editor, *Vistas in Information Handling*, volume 1, pages 1–29. Spartan Books, Washington DC, USA, 1963.

[40] D. Engelbart. Authorship provisions in Augment. In *Proceedings of the IEEE Compcon Conference*, San Francisco, USA, 1984. IEEE.

[41] G. Fitzpatrick, T. Mansfield, S. Kaplan, D. Arnold, T. Phelps, and B. Segall. Augmenting the workaday world with Elvin. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 431–450, Copenhagen, Denmark, Sept. 1999. Kluwer Academic Publishers.

[42] A. M. Fountain, W. Hall, I. Heath, and H. C. Davis. Microcosm: An open model for hypermedia with dynamic linking. In A. Rizk, N. Streiz, and J. Andrè, editors, *Proceedings of the European Conference on Hypertext*, Cambridge University Press, U.K., 1990.

[43] L. Fuchs. AREA: A cross-application notification service for groupware. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Copperative Work*, pages 61–80. Kluwer Academic Publishers, Sept. 1999.

[44] L. Fuchs, U. Pankoke-Babatz, and W. Prinz. Supporting cooperative awareness with local event mechanisms: The GroupDesk system. In *Proceedings of the 4^{th} European Conference on Computer Supported Cooperative Work*, pages 247–262, Stockholm, Sweden, 1995.

[45] R. Furuta, F. M. Shipman III, C. C. Marshall, D. D. Brenner, and H.-W. Hsieh. Hypertext paths and the World-Wide Web: Experiences with Walden's Paths. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8^{th} ACM Hypertext Conference*, pages 167–176, Southampton, UK, Apr. 1997.

[46] S. Goose, J. Dale, W. Hall, and D. D. Roure. Microcosm TNG: a distributed architecture to support reflexive hypermedia applications. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8^{th} ACM Hypertext Conference*, pages 226–227, Southampton, UK, Apr. 1997.

[47] K. Grønbæk. Eurocode workpackage wp2 task t2.2 report: Object oriented model for the distributed hypermedia toolkit. Technical Report CODE-AU-93-10, Aarhus University, Denmark, July 1993.

[48] K. Grønbæk, N. O. Bouvin, and L. Sloth. Designing Dexter-based hypermedia services for the World Wide Web. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8^{th} ACM Hypertext Conference*, pages 146–156, Southampton, UK, Apr. 1997.

[49] K. Grønbæk, J. A. Hem, O. L. Madsen, and L. Sloth. Designing Dexter-based cooperative hypermedia systems. In *Proceedings of the 5^{th} ACM Hypertext Conference*, pages 25–38, Seattle, USA, Nov. 1993.

[50] K. Grønbæk, J. A. Hem, O. L. Madsen, and L. Sloth. Cooperative hypermedia systems: A Dexter-based architecture. *Communications of the ACM*, 37(2):64–74, Feb. 1994.

[51] K. Grønbæk and J. Malhotra. Building tailorable hypermedia systems: the embedded-interpreter approach. In *Proceedings of the 9^{th} conference on Object Oriented Programming Systems, Language, and Applications (OOPSLA 1994)*, pages 85–101. ACM, 1994.

[52] K. Grønbæk and P. J. Nürnberg. Technical documentation for the COCONUT hypermedia services and infrastructure (Part 1). IM working papers 3, Intermedia, University of Aarhus, Denmark, 1999.

[53] K. Grønbæk and P. J. Nürnberg. Technical documentation for the COCONUT hypermedia services and infrastructure (Part 2). IM working papers 3, Intermedia, University of Aarhus, Denmark, 1999.

[54] K. Grønbæk, L. Sloth, and N. O. Bouvin. Open hypermedia as user controlled meta data for the Web. In *Proceeding of the 9^{th} World Wide Web Conference*, pages 553–566, Amsterdam, Holland, May 2000. W3C.

[55] K. Grønbæk, L. Sloth, and N. O. Bouvin. Open hypermedia as user controlled meta data for the Web. *Computer Networks*, (33):553–566, 2000.

[56] K. Grønbæk, L. Sloth, and P. Ørbæk. Webvise: browser and proxy support for open hypermedia structuring mechanisms of the World Wide Web. In *Proceedings of the 8^{th} International World Wide Web Conference*, pages 253–267, Toronto, Canada, 1999. W3C.

[57] K. Grønbæk and R. H. Trigg. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. In *Proceedings of the 7^{th} ACM Hypertext Conference*, pages 149–160, Washington DC, USA, 1996.

[58] J. M. Haake, T. Knopik, and N. Streitz. The SEPIA hypermedia system as part of the POLIKOM telecooperation scenario. In *Proceedings of the 5^{th} ACM Hypertext Conference*, pages 235–237, Seattle, USA, Nov. 1993.

[59] J. M. Haake and B. Wilson. Supporting collaborative writing of hyperdocuments in SEPIA. In *Conference proceedings on Computer-supported cooperative work*, pages 138–146, Toronto, Canada, Nov. 1992.

[60] F. G. Halasz. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31(7):836–852, 1988.

[61] F. G. Halasz, T. P. Moran, and R. H. Trigg. NoteCards in a nutshell. In *Proceedings of ACM Conference on Human Factors in Computing Systems and Graphics Interface*, pages 45–52, Toronto, Canada, Apr. 1987.

[62] F. G. Halasz and M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, Feb. 1994.

[63] W. Hall, H. C. Davis, and G. Hutchings. *Rethinking Hypermedia: The Micro-Cosm Approach*. Kluwer Academic Publishers, Norwell, USA, 1996.

[64] B. Halsey and K. M. Anderson. XLink and open hypermedia systems: A preliminary investigation. In *Proceedings of the 11^{th} ACM Hypertext Conference*, pages 212–213, San Antonio, USA, May 2000.

[65] C. Heath and P. Luff. Collaboration and control: Crisis management and multimedia technology in London Underground line control rooms. *CSCW Journal*, 1(1–2):69–94, 1992.

[66] J. Jühne, A. T. Jensen, and K. Grønbæk. Ariadne: A Java-based guided tour system for the World Wide Web. In *Proceedings of the 7^{th} International World Wide Web Conference*, Brisbane, Australia, 1998. W3C.

[67] C. J. Kacmar and J. J. Leggett. PROXHY: A process-oriented extensible hypertext architecture. *ACM Transactions of Information Systems*, 9(4):399–419, Oct. 1991.

[68] P. Kahn, J. M. Nyce, T. Oren, G. Crane, L. C. Smith, R. Trigg, and N. Meyrowitz. From Memex to hypertext: understanding the influence of Vannevar Bush. In *Proceedings of the 3^{rd} ACM Conference on Hypertext*, page 361, San Antonio, USA, Dec. 1991.

[69] R. Killough and J. J. Leggett. Hypertext interchange with the Dexter model: Intermedia to KMS. TAMU-HRL 90-002, Hypertext Research Lab, Texas A&M University, Aug. 1990.

[70] O. Lassila and R. R. Swick (editors). Resource Description Framework (RDF) model and syntax specification. W3C Recommendation 22 February 1999, W3C, Feb. 1999. `http://www.w3.org/TR/REC-rdf-syntax/`.

[71] J. J. Leggett and J. L. Schnase. Viewing Dexter with open eyes. *Communications of the ACM*, 37(2):77–86, Feb. 1994.

[72] G. Mark, N. A. Streiz, and J. M. Haake. Hypermedia use in group work: Changing the product, process, and strategy. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, (6):327–368, 1997.

[73] C. C. Marshall. Toward an ecology of hypertext annotation. In *Proceedings of the 9^{th} ACM Hypertext Conference*, pages 40–49, Pittsburgh, USA, 1998. ACM.

[74] C. C. Marshall and F. M. Shipman III. Spatial hypertext: Designing for change. *Communications of the ACM*, 38(8):88–97, 1995.

[75] H. Maurer. *Hyperwave — The Next Generation Web Solution*. Addison Wesley, 1996.

[76] M. Melly and W. Hall. Version control in Microcosm. In D. Hicks, A. Haake, D. Durand, and F. Vitali, editors, *Proceedings of the ECSCW'95: Workshop on the Role of Version Control in CSCW Applications*, number 289 in GMD-Studien, GMD-IPSI, Darmstadt, Germany, Apr. 1996.

[77] N. K. Meyrowitz. Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. In *Proceedings of ACM conference on Object Oriented Programming Systems, Languages and Applications (OOPSLA 86)*, 1986.

[78] N. K. Meyrowitz. The missing link: Why we're all doing hypertext wrong. In E. Barrett, editor, *The society of text: Hypertext, hypermedia and the social construction of information*, pages 107–114. MIT Press, Cambridge, USA, 1989.

[79] P. Mogensen and K. Grønbæk. Hypermedia in the virtual project room – toward open 3d spatial hypermedia. In F. M. Shipman, III, editor, *Proceedings of the 11^{th} ACM Hypertext Conference*, pages 113–122. ACM, May 2000.

[80] T. H. Nelson. Replacing the printed word. *Information Processing*, pages 1013–1023, 1980.

[81] P. J. Nürnberg. *HOSS: An Environment to Support Structural Computing*. PhD thesis, Department of Computer Science, Texas A&M University, College Station, Texas, USA, 1997.

[82] P. J. Nürnberg and H. Ashman. What was the question? reconciling open hypermedia and world wide web research. In *Proceedings of the 10^{th} ACM Hypertext Conference*, pages 83–90, Darmstadt, Germany, 1999.

[83] P. J. Nürnberg, J. J. Leggett, E. R. Schneider, and J. L. Schnase. Hypermedia operating systems: A new paradigm for computing. In *Proceedings of the 7^{th} ACM Hypertext Conference*, pages 194–202, Washington DC, USA, Mar. 1996.

[84] P. J. Nürnberg, E. R. Schneider, and J. J. Leggett. Designing digital libraries for the hyperliterate age. *Journal of Universal Computer Science*, 2(9):610–622, 1996.

[85] J. F. Patterson, M. Day, and J. Kucan. Notification servers for synchronous groupware. In *Proceedings of the Conference of Computer Supported Cooperative Work*, pages 122–129, Boston, USA, 1996.

[86] A. Pearl. Sun's link service: A protocol for open linking. In *Proceedings of the 2^{nd} ACM Conference on Hypertext*, pages 137–146, Pittsburgh, USA, Nov. 1989.

[87] T. A. Phelps and R. Wilensky. Robust intra-document locations. In *Proceeding of the 9^{th} World Wide Web Conference*, pages 105–118, Amsterdam, Holland, May 2000. W3C.

[88] W. Prinz. NESSIE: An awareness environment for cooperative settings. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Copperative Work*, pages 391–410. Kluwer Academic Publishers, Sept. 1999.

[89] O. Reinert, D. Bucka-Lassen, C. A. Pedersen, and P. J. Nürnberg. CAOS: A collaborative and open spatial structure service component with incremental spatial parsing. In *Proceedings of the 10^{th} ACM Hypertext Conference*, pages 49–50, Darmstadt, Germany, 1999.

[90] J. L. Schnase, J. J. Leggett, D. L. Hicks, P. J. Nürnberg, and J. A. Sánchez. Design and implementation of the hb1 hyperbase management system. *Electronic Publishing — Origination, Dissemination and Design*, 6(1):125–150, 1993.

[91] J. L. Schnase, J. J. Leggett, D. L. Hicks, P. J. Nürnberg, and J. A. Sánchez. Open architectures for integrated, hypermedia-based information systems. In *Proceedings of the 27^{th} Hawaii International Conference of Systems Sciences*, Maui, Hawaii, USA, Jan. 1994.

[92] N. A. Streitz, J. Geißler, J. M. Haake, and J. Hol. Dolphin: integrated meeting support across local and remote desktop environments and liveboards. In *Proceedings of the conference on Computer supported cooperative work*, pages 345–358, Chapel Hill, USA, Oct. 1994.

[93] N. A. Streiz, J. M. Haake, J. Hannemann, A. Lemke, W. Schuler, H. Schütt, and M. Thüring. SEPIA: A cooperative hypermedia authoring environment. In *Proceedings of the European Conference on Hypermedia Technology (ECHT 1992)*, pages 11–22, Milan, Italy, 1992.

[94] R. H. Trigg, L. A. Suchman, and F. G. Halasz. Supporting collaboration in NoteCards. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 153–162, 1986.

[95] E. J. Whitehead Jr. An architectural model for application integration in open hypermedia environments. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8^{th} ACM Hypertext Conference*, pages 1–12, Southampton, UK, Apr. 1997.

[96] E. J. Whitehead Jr. Control choices and network effects in hypertext systems. In *Proceedings of the 10^{th} ACM Hypertext Conference*, pages 75–82, Darmstadt, Germany, Feb. 1999.

[97] E. J. Whitehead, Jr., K. M. Anderson, and R. N. Taylor. A proposal for versioning support for the chimera system. In *Proceeedings of the Workshop on Versioning in Hypertext Systems. Held in connection with ECHT '94*, pages 45–54, Edinburgh, Scotland, Sept. 1994.

[98] E. J. Whitehead Jr. and Y. Y. Goland. WebDAV: A network protocol for remote collaborative authoring on the Web. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 291–310, Copenhagen, Danmark, 1999. Kluwer Academic Publishers.

[99] U. K. Wiil. Issues in the design of EHTS: A multiuser hypertext system for collaboration. In *Proceedings of the 25^{th} IEEE Hawaii International Conference on System Sciences*, volume 2, pages 629–639, 1992.

[100] U. K. Wiil. Experiences with HyperBase: A multiuser hypertext database. *ACM SIGMOD RECORD*, 22(4):19–25, Dec. 1993.

[101] U. K. Wiil. Hyperform: Rapid prototyping of hypermedia services. *Communications of the ACM*, 38(8):109–111, 1995.

[102] U. K. Wiil and J. J. Leggett. The HyperDisco approach to open hypermedia systems. In *Proceedings of the 7^{th} ACM Hypertext Conference*, pages 140–148, Washington DC, USA, Mar. 1996.

[103] U. K. Wiil and J. J. Leggett. Workspaces: The HyperDisco approach to Internet distribution. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8^{th} ACM Hypertext Conference*, pages 13–23, Southampton, UK, Apr. 1997.

[104] U. K. Wiil and P. J. Nürnberg. Implications of open hypermedia systems research for the World Wide Web. In *Proceedings of the 3rd IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA '99)*, pages 296–304, Nassau, Bahamas, Oct. 1999.

[105] U. K. Wiil, P. J. Nürnberg, D. Hicks, and S. Reich. A development environment for building component-based open hypermedia systems. In F. M. Shipman, III, editor, *Proceedings of the 11th ACM Hypertext Conference*, pages 266–267. ACM, May 2000.

[106] N. Yankelovich, B. J. Haan, N. K. Meyrowitz, and S. M. Drucker. Intermedia: the concept and the construction of a seamless information environment. *Computer*, pages 81–96, Jan. 1988.

Unifying Strategies for Web Augmentation

Niels Olof Bouvin

Aarhus University,
Department of Computer Science,
Aabogade 34A, DK8200 Aarhus N, Denmark

bouvin@daimi.aau.dk

ABSTRACT

Since the beginning of the WWW, tools have been developed to augment the functionality of the Web. This paper provides an investigation of hypermedia tools and systems integrating the World Wide Web with focus on functionality and the techniques used to achieve this functionality. Similarities are found and based on this, a new framework, the Arakne framework, for developing and thinking about Web augmentation is presented. The Arakne framework is flexible and supports most kinds of Web augmentation. Finally an implementation of the Arakne framework is described and discussed.

KEYWORDS

Web Integration, Open Hypermedia Systems, Open Hypermedia Protocol, Collaboration on the Web, Unifying interfaces, Common Reference Architecture for open hypermedia systems, Java

INTRODUCTION

The World Wide Web has in an amazing short time span become the hitherto largest hypertext and is pervasive in everyday life as few things before. This success has in part been attributed to the simple architecture behind the Web: A stateless file transfer protocol (HTTP), an universal Internet naming scheme (URL) and an easily understood document format (HTML). These standards have (largely) been adhered to, and this has enabled the creation of a large amount of software, be it Web servers or browsers to work together to the benefit of all.

The success of this simple and hugely scaleable architecture has come, from the standpoint of the hypermedia research community, at some costs, as the Web itself is lacking in the ways of more advanced (but ironically often far older) hypermedia systems. Web links are unidirectional jump links, embedded in the HTML documents, severely diminishing the flexibility of use. There is yet no widespread support for collaborative authoring, though an initiative such as WebDAV [13]) holds great promise.

An important development in the hypermedia community in the nineties has been the focus on and development of open hypermedia systems integrating third-party applications. Significant work has been done in systems such as MicroCosm [9][20], HyperDisco [44][45], HOSS [32], DHM [14][15], and Chimera [1]. These systems have all addressed the problem of augmenting third-party applications, and the lessons learned are important guidelines for future work of Web augmentation. Based on these experiences,, taxonomies and studies have been developed, by Whitehead [43], Grønbæk & Wiil [17], and Wiil & Østerbye [46][48], to help researchers and developers discuss and reason about the open hypermedia field and how to utilise hypermedia in third-party applications. Given this experience of integration it should come as no surprise that the open hypermedia community was quick to develop open hypermedia Web integrations, e.g. DLS [7], DHM/WWW [16], and Chimera [2]. This article will investigate how they and others achieved their goals of Web augmentation.

Adhering to standards has a large part of the success of the Web and the very size of the Web has enormous inertia, so an attempt to replace the Web with something perhaps more advanced in certain aspects is, if not doomed, then up against tremendous odds. Clearly this is not the way to improve the Web. An approach that retains the benefits of the Web as well as adding new desired functionality is the development of systems that operate within Web standards, be it HTTP, HTML, browsers, or servers. This is in spirit with Meyrowitz' call for hypermedia integration in third-party applications [27], and the amount and quality of work done using this approach would suggest that it is at the very least possible.

AN OVERVIEW OF WEB AUGMENTATION STRATEGIES

As the scope of this article is to study the techniques used in Web augmentation tools, an overview of augmentation approaches is in order. This overview will work at two levels of abstraction: first a broad grouping based on the functionality of the tools and later a more specific characterisation of the individual tool. This characterisation will be based on the chosen approach to common problems, such as storage, Web browser integration, level of support for collaboration, and so forth. The former level of abstraction will let us discuss and compare related tools, while the latter will allow us to recognise reoccurring themes in the approaches taken.

A tool shall be considered a Web hypermedia augmentation tool, if it through integration with a Web browser, a HTTP proxy or a Web server adds content or controls not contained within the Web pages themselves to the effect of allowing structure to be added to the Web page directly or indirectly, or to navigate such structure. The purpose of such

Hypertext 99 Darmstadt Germany

a tool is help users organise, associate, or structure information found on the Web. This activity may be done by a single user or in collaboration with others.

Following this definition, the purpose of the Web augmentations reviewed for this article is to help users structuralise their Web work. Either by adding structure and displaying it, or by extracting structure already present and making it more visible. The displayed structure may be malleable, allowing the user to modify it. The Web augmentation tools reviewed for this article have been divided into four categories:

- Annotations/Discussion support
- Link creation and transversal
- Guided tours
- Structuring/Spatial

We claim no universality to this categorisation, but have found it handy when discussing the Web augmentation tools and their use.

A Web augmentation tool can be classified by the schema summarised in Table 1. Thus a tool can either be a part of a Web browser; it can be an closely integrated tool loaded on runtime; it can reside temporarily within the browser as an applet or an ActiveX control; it can be an application running the user's computer; or it can be located elsewhere (in that case more often than not as a HTTP server or proxy).

The Web augmentation tool can either have no storage requirements; it can store it data locally on the host computer, or remotely on a server.

Many of these Web augmentation tools modify Web pages, either to insert interfaces of their own or to add structure (e.g. links) to the Web page. This modification can take place at the Web server (perhaps a Web server translating a proprietary data format into HTML), by calling CGI-scripts that return modified pages, by using a special proxy, or by modifying the Web pages as or after they are displayed in the Web browser.

As for the collaborative aspect, a Web augmentation tool can either be strictly personal, i.e. relevant to a single user only; the created data or structures can be shared (e.g. send to another user or placed on a Web site); the structures can browsed or edited by turn-taking (asynchronously); or users can collaborate through the structures in real time (synchronously).

WEB AUGMENTATION TOOLS

In this section we will describe various Web augmentation tools focusing on the elements introduced by Table 1. The scope of this article does not allow for a comprehensive study nor a general survey, so only a subset of the existing Web augmentation tools is presented.

Annotations/Discussion support

Bush envisioned marginalia in the Memex [5], and the interest in annotations and how they should be supported by hypermedia has not diminished over the years, as witnessed by the investigation done by Marshall [25].

The first widely successful Web browser, NCSA Mosaic [30], gave the user the opportunity to create annotations to Web pages. The annotations were personal and stored locally. Later, this feature fell out of favour with Web browser developers.

Recognising that annotations whilst useful for the individual are even more beneficial for a community, several collaborative annotation tools have been created. Röscheisen *et al.*[35][36] have developed ComMentor, which have used for several purposes, including content rating and annotations. The system employs a Mosaic [30] browser, modified to provide an interface to the annotation server, which consists of a collection of CGI-scripts. The user has alongside with the browser a merge library, which inserts comments and links to comments into the Web pages. Annotations are stored in sets, of which the user may activate an arbitrary

Method of integration
A part of the browser
Browser add-ons
Within the browser (plug-ins, applets and JavaScript)
Without the browser, local
Without the browser, remote
Within the proxy
Within the Web server
Location of storage
No storage
Local storage
Remote storage
Web page modification
No modification
At the Web server
CGI-scripts at a Web server
At the proxy
In the browser
Level of collaboration supported
Personal
Shareable
Asynchronous collaborative
Synchronous collaborative

Table 1 - A classification scheme of Web augmentation tools

number. Collaborative annotation is supported through dividing users into groups that may share sets of annotations. A set of annotations can be set to be private, available to a group, or publicly available. Annotations are displayed using in-place markers (small pictures) indicating either the nature of the annotation or the author's identity. An annotation while write-protected may also be annotated.

Another system is CritLink Mediator, part of CritSuite [10], which is specialised to provide support for 'critical discussions'. CritLink employs predefined typed links (support, issue, comment, query) akin to many hypermedia systems, such as IBIS [34] and TEXTNET [40]. The comments are created by a mix of CGI-scripts, Web forms, and JavaScript and stored either on a designated server or in the user's own Web space as ordinary Web pages. Links to the comments

are inserted into Web pages by use of a Web server effectively acting as a proxy server. The pages are also modified to include a tool bar used for navigation, annotation, and the launch of CritMap [39], a tool that generates maps of neighbourhood Web pages. Links in pages presented by the server are modified to go through the server. As comments are Web pages in their own right, they can also be commented on. There are only one set of annotations and no notion of groups, though the individual user is identified.

Creating links

As noted in the introduction, quite a few research projects have addressed the issue of adding external structures to Web pages. An excellent investigation of the various approaches taken by the open hypermedia community can be found in [2]. There are two main approaches: links (or other kinds of structural information) are either displayed alongside the Web page or inserted into the Web page. The former case requires either a program to display this structure or a browser window where the structure has been converted to HTML. The latter case involves modifying HTML pages on the fly, which can be done at three places: at the origin, in transit or at arrival, i.e. the Web server, the HTTP proxy, or the Web browser.

Chimera [1][2] is an example of a system, where experiments with either displaying structure information in a separate program (an applet) or making the structure server accessible through HTTP have been carried out. By modifying a Web server to interpret HTTP requests as requests to a Chimera server to which the Web server is hooked up and in turn translating the Chimera structures to HTML, a user is able to browse the hypermedia structure using an ordinary Web browser. This experiment was extended upon by the creation of a Java applet capable of displaying Chimera hypermedia structures. By the combination of a special Web server, CGI-scripts and cookies, this applet was inserted into all pages displayed in the Web browser, giving the user immediate access to Chimera services.

Hyper-G [26] is a more specialised system, as it to achieve full functionality relies on a special document format (HTF – Hypertext Format), a special server, and a custom browser. It is however possible to interface to the system using an ordinary Web browser using a special WWW-gateway, that will translates HTF document and hypermedia structure to HTML. The hypermedia system offers strong support for hierarchical structures and searching, and allows users without a special browser to create links using forms. In recent versions HyperWave [22] (as the system is now known) offers an advanced interface utilising Java-applets and JavaScript inserted into Web pages by the HyperWave server.

DLS [6] (Distributed Link Service) is based on the MicroCosm hypermedia system [7][9][20]. The first DLS systems used a wrapper to attach a link service menu to a browser (this integration being dependent on whether an integration existed for the user's browser), thus creating a (in the terminology of Whitehead in [43]) shim integration with a third party application. Links were followed by selection of text and selecting 'Follow Link' in the attached menu. This would cause the wrapper to contact the link server with an URL encoding the request, resulting in a Web page of the matching links. To address the problem of having to install special software and to make links more visible, an interfaceless version was developed that used a link server proxy to insert links in Web pages as requested by the user. The user used a form to configure which link bases to use and how the link should be presented in the document (to make the distinction between links belonging in the document and inserted links clear). Due to performance issues (beyond 'conventional' links, MicroCosm offers computation intensive links, such as keyword links, person links and citation links) and copyright and authors' rights concerns about adding content to Web pages, a new design was introduced with the AgentDLS [8]. Rather than offering synchronous links (presented together and simultaneously with the document), links are now displayed in a separate window. This improves the performance of browsing considerably, as the users' primary window of interest does not have to wait for links to be resolved. The linking service thus takes on a more advisory nature. This system is implemented by using a proxy that (as seen by the Web browser) acts as a normal proxy but also sends the displayed document to a link server agent that resolves the links relevant to the document. The display of these links is handled by having the AgentDLS browser window request a page from the link server agent with regular intervals.

The Devise Hypermedia group, of which the author is a member, has also made various Web integrations with its Dexter-based hypermedia system [14][15]. The first attempt was DHM/WWW [16]. The architecture consisted of a Java-applet communicating through a CGI-script to a DHM server. When the user requested a document by typing its URL in the applet, the applet would retrieve the document while querying the DHM server for endpoints in the document. The endpoints retrieved was inserted into the Web page as it was being downloaded and displayed in a Web browser window using JavaScript. All links in the Web page were modified so that a click on a link would result in the applet being invoked, allowing it repeat the above described process. The links and endpoints from the DHM server could also be inspected and browsed within the applet. This version had several shortcomings: it was dependent on the user not using bookmarks or entering URLs in the Web browser itself, as such actions would cause the applet to be terminated as its own page would be unloaded. Furthermore it was unable to handle frames (as the loading of a new 'top' frame set would also cause the unloading of the applet) and was limited by the 'sandbox' imposed for security reasons on Java applets[1]. While supported by the DHM server, the DHM/WWW applet could only handle one context (that is one set of hypermedia structures) and had no user concept. A second version, Navette [3], was developed to address some of these issues. Navette was a signed Java applet, allowing the system to use Web pages from arbitrary Web

[1] The Java 'sandbox' security limits a Java applet in various ways. Most crucial to DHM/WWW was the restriction of network contact exclusively to the originating web server, thus making DHM/WWW unable to work with web pages from other web servers.

servers. To speed up communication with the DHM server, TCP/IP and optimistic caching of hypermedia structures (e.g. retrieving a whole context rather than only resolving one link) were used. This version also handled multiple contexts and users. The frame problem remained, and Web pages were still displayed using JavaScript, which made for noticeable degraded performance when browsing with Navette. Simultaneously with Navette, the Webvise client [18], a custom integration with the Microsoft Internet Explorer [28] was being developed. Operating as an application rather an applet removed the limitations put on DHM/WWW and Navette, and using the Microsoft Internet Explorer [28] rather than the Netscape Communicator [31] allowed Webvise to insert links after the browser had displayed the document, thus improving performance considerably. This is done through DOM [11] and the COM-interface available through the Internet Explorer. A second version of Navette has been developed addressing the problems of prior releases using the Arakne framework, which will described below.

Guided tours

Guided tours and trails have been a part of hypermedia from the very beginning [5], when Bush introduced the concept of the trail linking related documents together. Trigg did more recent groundbreaking work in [41]. Several existing systems try to exploit this idea with Web documents.

Walden's Paths [12][37] is a system designed mainly to be used in an educational setting, where a teacher composes trails for students to follow. The teacher uses either the Path Authoring Tool (a Java application) or VIKI [38] combined with a browser as an authoring tool. Trails are stored on a Path Server, which through the use of CGI scripts acts as a proxy while modifying the pages to provide an interface to the path. The interface consists of blocks in the top and the bottom of the page. This block is a graphical representation of (a part of) the path plus additional annotations written by the path author. As all documents go through the Path Servers (links in the documents are modified to achieve this), students can go 'off path' and still return to the path by pressing a button in the interface block. State is communicated by adding arguments into the URL given to the Path Server's CGI-scripts. The Walden's Paths has been extended with regards to collaborative aspects, allowing students to author and share paths of their own. Additionally work has been done to extend upon the linear path by adding conditional branches. The logic to support this is handled by the Path Server, thus still making all functionality accessible from a standard Web browser.

Another Web-based guided tour system is Ariadne [23], which is a Java-based applet. Ariadne operates in an external window to the browser and controls the browser through JavaScript. A guided tour in Ariadne is a directed graph, as opposed to the linear (with branches) path of Walden's Paths. The Ariadne user interface supports both browsing and editing of guided tours. The tours are stored as composites on the Dexter-based DHM [15] server. Leaving the Web pages untouched has several advantages to the Walden's Path approach, as 1) it reduces overhead and complexity as Web documents do not have go through an extra server, and 2) entering URLs or using bookmarks does not pose a problem. On the other hand users are required to use a Java-enabled Web browser rather than any Web browser, though that currently is not a strong requirement. The Ariadne system has recently been adapted to work within the Arakne framework, which will be described in more detail below.

Structuring/Spatial

Spatial hypermedia as described by Marshall & Shipman [24] and as implemented in VIKI [37] is a new kind of hypermedia application, where link structures are no longer explicit but rather implicit based on the spatial relationship between objects. This has become a very powerful tool for organising and structuring, and few hypermedia systems are in more need of organisation and structure than the Web.

Web Squirrel [42] is a URL management system, that uses a spatial metaphor to help users organise their URLs into 'information farms'. The user creates Neighbourhoods onto which URLs are dragged and dropped. The Neighbourhoods and the URLs are arranged spatial as the user wishes, and are analysed by software agents that can create links between URLs according to user's rating of the Web sites and maintain link integrity. The user can create new agents using a scripting language. The information farms are stored locally, but can be distributed to other users of Web Squirrel, exported as HTML, or converted to the Hot Sauce MCF format.

Hot Sauce [21] is a spatial hypermedia plug-in created by Apple. Hot Sauce displays a zoomable 2D representation of a collection of collections and documents. This structure is stored using the XML [47] based Meta Content Format [19], a general format to describe meta content (MCF has thus much wider application than its use in Hot Sauce). Links and collections of links are arranged spatial and the user can zoom in and out, move about, and open collections within the collection. If the user double-clicks on a link, the document is retrieved in a separate window, allowing the user to continue to navigate using Hot Sauce. The Hot Sauce is a media viewer and as such retrieves its MCF file from a Web server.

SUMMARY OF STRATEGIES FOR WEB AUGMENTATION

The tools and systems described above have addressed many problems pertaining to the current Web, and have utilised a lot of different techniques to attempt to solve these problems. The Web augmentation strategies are, using the schema introduced in Table 1, summarised in Table 2. We will below outline some general trends and describe some of the aspects of writing Web augmentation tools that make developing them hard.

Some patterns become apparent. All of the tools reviewed are responsive and need to be aware of the user's actions, be it to record the URL of the current Web page or to perform link and endpoint computations. All need to provide the user with a user interface (though it might be very discreet at most times, as in the 'interfaceless' DLS). Most of the tools need to store data somewhere and most choose to do this on a remote server, thus raising the need to able to communi-

cate over network. The communication is often handled by CGI-scripts, which is problematic, as the tool only gets data when it requests it – the server cannot notify the tool of changes. Many of the tools modify Web pages, and most of the implementations of this functionality would have a hard time interoperating with each other, as they in turn would modify pages and links, quite possibly corrupting each other's data. Most of the Web page modifications are not robust to things such as frames and JavaScript, and many have a problem with forms. The tools relying on modifying link with CGI-scripts rather than using a proxy are fragile to the use of bookmarks or directly entered URLs. The tools that modify Web pages through a proxy are hard to use with other tools that rely on proxies to be informed of the user's actions, unless either of the proxies can it be modified to use the other as a proxy. Furthermore the use of proxies requires the user to modify the Web browser configuration which can be unwieldy if the user does not wish to continually use the tool relying on the proxy. These problems to which there generally are no easy solutions, make it difficult for the developer to create Web augmentation tools.

Tool	Method of Integration	Storage	Web page modification	Collaboration support
ComMentor	Part of browser, CGI-scripts	Remote	Local proxy	Asynchronous
CritLink Mediator	Within browser, JavaScript, forms; CGI-scripts	Remote	CGI-scripts	Asynchronous
Chimera	Web server	Remote	No	Asynchronous
Chimera	Within browser, applet; Web server; cookies, CGI-scripts	Remote	Web server	Asynchronous
Hyper-G	Part of browser/Within browser, forms; web-server;	Remote	Web server	Asynchronous
HyperWave	Within browser, applet, JavaScript, forms; web-server	Remote	Web server	Asynchronous
DLS	Without browser, local; within browser, forms	Remote	No	Asynchronous
DLS	Within browser, forms; proxy	Remote	Proxy	Asynchronous
AgentDLS	Within browser, separate window; proxy	Remote	No	Asynchronous
DHM/WWW	Within browser, applet, JavaScript, CGI-scripts	Remote	Web browser	Shared
Navette	Within browser, applet, JavaScript	Remote	Web browser	Asynchronous
Webvise	Without browser, local	Remote	Web browser	Asynchronous
Walden's Paths	Within browser, JavaScript, forms; CGI-scripts	Remote	CGI-scripts	Shareable
Ariadne	Within browser, applet, JavaScript	Remote	No	Asynchronous
Web Squirrel	Without browser, local	Local	No	Shareable
Hot Sauce	Within browser, plug-in	Remote	No	Shareable

Table 2 - Summary of Web augmentation strategies

TOWARDS A COMMON FRAMEWORK

We are aware of no single "silver bullet", or all encompassing solution, that will solve all the problems described above. However, the similarities between the described Web augmentation tools would suggest that it should be possible to describe and model their functionality in a common framework. This could provide workers in the field with a tool for future conceptual and practical development. Trying to create such a tool, we have come up with the Arakne framework.

The Arakne framework is a conceptual model, which has been implemented as an environment for Web augmentation tools. The implementation is just one implementation of a general framework. The practical issues raised by the summary above will be dealt in the description of the implementation.

The Arakne framework is an object-oriented, component-based, three-layered model aimed at providing Web augmentation tools a unified access to structure servers, proxies, and Web browsers. It is an instantiation of the Common Reference Architecture (CoReArc) for open hypermedia systems, as described by Grønbæk & Wiil [17]. CoReArc divides the architecture of hypermedia systems into three layers: The content layer (displaying and handling documents, displaying structure), the service layer (handling navigation, integration, collaboration etc.), and the structure layer (storage and retrieval of structure).

The Arakne framework is aimed at modelling Web augmentation tools, and the elements contained in the model should now be familiar.

A diagram of the framework can be seen in Figure 1. The framework may support any number of Web augmentation tools. These tools (known as 'navlets') are dependent on four core components of the Arakne framework: the Operations, the Hyperstructure Store, the Browser, and the Proxy. The navlet is the domain specific part of a Web augmentation tool. It provides a user interface as well as special logic to handle the specific domain. This may include deciding which links to display in a Web page based on information retrieved from the Hyperstructure Store component, or interfacing to the Proxy component for analysis of documents. Depending on the situation the computation and analysis may be carried out by the navlet or by another component.

The Operations component models the communication with the structure server layer. This component will thus typically support the same services as the structure server(s). This is where on the wire issues, such as network communication, marshalling, and multiplexing, are handled.

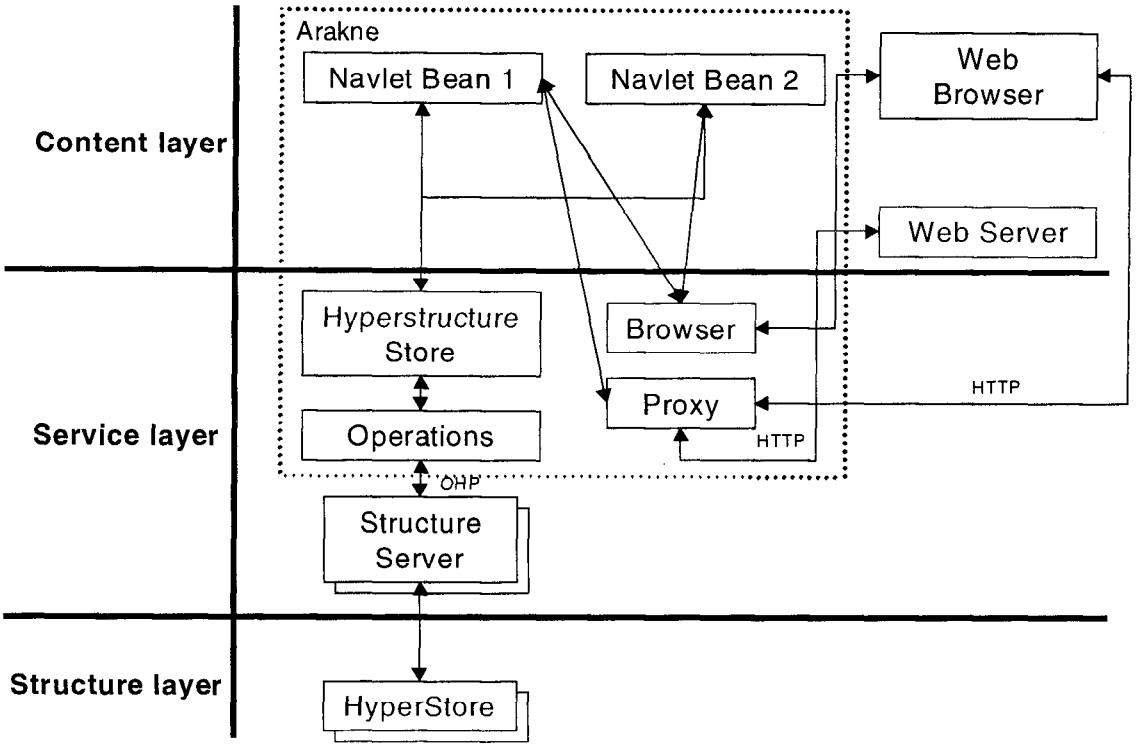


Figure 1 – The Arakne Framework

The Hyperstructure Store is the interface between the navlets and the Operations. The Hyperstructure Store provides convenience functions for the navlets, as well as caching the results of the queries retrieved with Operations. The Hyperstructure Store will also alert navlets to changes in the structures they subscribe to.

The Proxy component models the modification and analysis of Web content. Depending on their domain, navlets may require the Proxy to modify Web pages, and these requests for modifications are collected by the Proxy and used to modify the Web page. Other navlets may require access to the content of a Web page and the Proxy also handles this.

The Browser component models the user's Web browser. Through the Browser navlets can retrieve and modify the state of the Web browser such as which URL is currently displayed; the structure of the current frame set; whether a selection has been made in a frame and if so, what and where.

The situation depicted in Figure 1 is a situation of two navlets, where Navlet Bean 1 is a link creation tool, and thus needs access to the Proxy in order to insert links into Web pages. Navlet Bean 2 is a guided tour tool and does not modify Web pages; and is not connected to the Proxy. Both however need to be able to tell and set the state of the currently displayed documents, as well as retrieving data from the structure server through the Hyperstructure Store.

Mapping Web Augmentation Tools to Arakne

If we review the models of the Web augmentation tools described in this paper, most can be mapped to the abstract Arakne framework. We will in this section argue for the general applicability of the Arakne framework in each of the general Web augmentation categories introduced earlier, and specify the mapping of one representative from each category to the Arakne framework.

Annotation tools are in their functional requirements similar to link creation tools and will dealt with as one. Both need to retrieve hypermedia structures stored on a server, which are handled by the Operations and the Hyperstructure Store components. Many of these tools need to modify Web pages in order to insert links or annotations, which is handled through the Proxy component. Support for collaboration is handled at different levels. The Hyperstructure Store component is able to handle sets of structures as well multiple users. Notifications from the structure servers are handled by the Operations component and forwarded to the Hyperstructure Store component, which notifies navlets, depending on what events they subscribe to.

The ComMentor system, described in [35], has three main elements apart from the structure servers and the Web servers; namely the merge library, the interactive renderer, and the user context control application. The merge library handles the tasks of the Operations, Hyperstructure Store, and Proxy components. The modifications made to the Mosaic browser combined with the code handling it in the user context control application makes up the Browser component.

The AgentDLS [8] architecture consists of a link server agent, that analyses Web pages sent to it by a HTTP proxy and based on this uses link resolvers and link bases to com-

pile a list of relevant links. The role of the proxy is a simple case of the Proxy component with the twist that documents are sent to two parties. The link server agent acts as a combined Operations and Hyperstructure Store component. The analysis performed by link resolvers also would seem to belong to this component, depending on the set-up of the navlet. The Browser component's responsibility – letting the system set or get the state of the Web browser – is handled at two levels: in the Proxy component (what document is displayed now?) and in the AgentDLS window, where users can directly influence, what URL to display in their Web browser.

Guided tour tools require a structure server for storage of the tours as well as the ability to communicate with this server. This can be modelled through the Operations and Hyperstructure Store components. To keep track of where the reader is on the trail (and to put the reader back on track, if need be), it must be possible to set and read the state of a Web browser, which is modelled through the Browser component. Some guided tour tools, e.g. Walden's Paths, modify the Web pages to add comments and interface. While it may not be necessary for interface reasons in the Arakne framework (this could be handled by a navlet), it certainly can be done through the use of the Proxy component.

Could the current implementation of Walden's Path be fitted within the Arakne model? In this case, some of the Arakne components collapse into one. The Path Server acts as the Proxy, Hyperstructure Store, and Operations components, by handling both external structures (guided tours) and Web page modification. The Browser component aspect is handled by the modification of links that keeps the system (i.e. the Path Server) aware of the state of the Web browser. The interface aspects of the navlet are the header and footer of the displayed Web pages that give the user a possibility to interact with the system.

Spatial and structuring Web augmentation tools certainly need structure servers and information about the currently display Web page just as the rest of the above described tools. However, the main focus of the spatial/structure tools described in this paper has been on the user interface and the ability to analyse and process structure and relationships using user written scripts. The user interface is handled by the navlet itself. While the navlet of course would serve as the front-end, the scripting capability fits best within the Hyperstructure Store component, where all structure is stored during run-time and where the Operations component can be directly accessed.

AN IMPLEMENTATION OF THE ARAKNE FRAMEWORK

A system called the Arakne applet has been developed as a part of the Coconut project. The immediate goal of the implementation was to try out the soundness of the Arakne framework by integrating the guided tour tool Ariadne and the link creation tool Navette. The system has undergone some development and has proven robust to the interchange of components.

The system was originally developed as a Java applet running in Netscape Communicator [31]. The components within the dotted line in Figure 1 were a part of this particular implementation. This was an implementation choice; as can be seen from the previous section, the framework itself posits no such requirements on the location of the individual components.

The components in Figure 1 correspond to Java classes in the Arakne applet. The interfaces between components are handled in the MVC (Model View Control) idiom, where the 'view' (e.g. a graph displaying a guided tour) is loosely coupled through events to the actual data, the 'model'. Modifications of data is handled as requests through the 'controller' and if granted notified to the view through the use of events.

The Hyperstructure Store class caches structure retrieved through the Operations class and is the only interface to the structure servers given to the navlet beans. This is done to improve performance (by not retrieving the same information twice) and to ensure data integrity between the Hyperstructure Store and the structure servers. The Hyperstructure Store provides the navlet beans with a rich set of convenience methods, which are translated into queries to the structure servers by the Operations class.

The Browser class encapsulates the needed functionality to communicate with a Web browser, in the original case the Netscape Communicator. The interface is relatively simple and can be adapted to another browser.

The Proxy class is a small proxy running as a thread in the applet. The Netscape Communicator can via JavaScript be set up to use a proxy on the fly, and when the Arakne applet is running, the Proxy class acts as a proxy for the browser. The Proxy class uses whatever proxy the browser was configured to use, and when the Arakne applet is terminated, everything is returned to normal. The Proxy handles requests for Web page modifications and the correct display of frames (thus allowing user to link into frame sets).

All of the classes visible to the navlet beans, the Hyperstructure Store class, the Proxy class, and the Browser class generate events that the navlet beans may subscribe to. The navlet beans are currently restricted to being Java beans and to follow some design guidelines. The current implementation supports only compile time integration of navlet beans giving the user a fixed number of available navlet beans, but future versions of the Arakne applet should be able to load navlet beans on runtime so that the user can retrieve navlet beans from different servers.

The current version of Netscape Communicator[2] has a very limited API for browser integration as well as a faulty Java socket implementation. This lead to the abandonment of the Communicator as the supported Web browser at a time where most components as well as two navlets were finished or nearing completion. The Microsoft Internet Explorer was chosen as the new supported Web browser. This choice has brought some costs, as the Arakne applet is now not only browser-dependent (which was also the case before), but

[2] Version 4.5

now also platform dependent, as Java is not supported in the Microsoft Internet Explorer on anything but the Windows platform.

The migration to the Internet Explorer also served as a (unintended) test of how much code needed to be rewritten to accommodate the change. This code rewrite has been light. The Browser class has been redone, as the Internet Explorer is controlled not through JavaScript, but through a COM interface. Furthermore some unique features such as the ability to operate on selected text in a browser window through the use of right-click menus have been exploited in some changes in user interface, e.g. link creation.

The Internet Explorer does not allow proxy configuration through JavaScript or other means, which has led to some major changes in the Proxy component. The current version relies on the DHMProxy [18] for Web page modification. This is expected to change as the DOM model [11] gives excellent possibilities for Web page analysis and modification, after the Web page has been rendered by the Web browser. This work is expected to be heavily dependent on the experiences from the Webvise application [18].

The current version uses the Devise Hypermedia server as a structure server. This is changing rapidly however, as a new Hyperstructure Store class is being developed to use a new Operations component that utilises the Open Hypermedia Protocol [33]. This has numerous advantages. The OHP is a powerful protocol with a good and general data model. Among the interesting features of OHP is the support for sessions, so that e.g. link following happens simultaneously on several machines as demonstrated at the demo at Hypertext 98. Finally the open hypermedia community supports the OHP and the data model, so that the Arakne applet may communicate with other structure servers such as OHP compliant MicroCosm or HyperDisco servers.

The synchronous collaborative aspects of the Arakne applet have been put on hold until the OHP is fully integrated, though the system as it is supports asynchronous collaboration.

Currently the Arakne applet supports the Ariadne and Navette navlet beans. The user is thus able to create links in documents while creating guided tours, which was not possible before. Each of the navlet beans occupies an internal window in the Arakne applet. The Arakne applet provides the interface for logging on to structure servers and the selection of contexts.

The Navette navlet has recently been extended to support linking to and from segments in video and audio clips through the Mimicry system [4]. Most temporal media plugins do not have APIs well suited for the needs of open hypermedia, which has led to the development of the Mimicry player. The Mimicry player is a Java applet capable of handling most video and audio formats, and it provides a rich API for open hypermedia integration.

Future Plans

Future plans for the Arakne applet include more navlet beans. Currently a spatial navlet bean is under development. The navlets currently supported are not aware of each other, but future versions of Arakne will support inter-navlet communication. The Microsoft Internet Explorer is not expected to be the only Web browser supported by future Arakne applets. The upcoming Mozilla 5.0 holds great promise in this regard. Another area of interest to the developers is the implications of XLink and XPointer.

Experiences of the Development of the Arakne Applet

How does the current Arakne applet compare to the problems raised in the summary of strategies for Web augmentation? Network communication is handled through sockets by the Operations Component, and the client maintains a socket connection, so that the structure servers may contact it. The Arakne applet has so far only supported navlets orthogonal to each other, so there is currently no experiences with regards to two navlets trying to modify the same Web page. However the architecture has only one component, the Proxy component, allowed to modify Web pages, so it should be possible to contain and perhaps avoid most problems. The Arakne applet is currently dependent on a proxy with the usual advantages (all browsing activities are 'captured' by the system) and disadvantages (the user needs to configure the Web browser to use the proxy). The proxy can use other proxies without problems, though the use of two Web page modifying proxies certainly would lead to undefined results. The proxy handles frame sets and inserts links through the use of JavaScript and DOM, so that Web pages are (visibly) modified on arrival rather than in transit, which solves many problem regarding dynamically created documents and frame sets. The Arakne applet is a Java applet with the limitations imposed on Java applets. The security restrictions are handled through digital signing. The Arakne applet is fairly secure from being unloaded by mistake, as it runs in a separate non-resizable browser window with disabled menus and toolbars. A consequence of the integration with the Microsoft Internet Explorer is that a user can not just download and use the Arakne applet. Certain files have to be installed by the user to provide the Arakne applet with right-click menus. This process can however be largely automated.

Some of the solutions found in the Arakne applet are strictly browser and platform dependent. This is very unfortunate, and the only thing that can remedy the situation is a new standard Java API for Web browsers. This API should at least provide functionality similar to that of the API of the Microsoft Internet Explorer, but do so in a browser independent way. Given the current Web browser situation, we think that such an API is unlikely to appear anytime soon, so the next best solution would be a Web browser that provided a platform independent API. Whether or not Mozilla 5.0 will provide such an API remains to be seen.

CONCLUSION

Web augmentation tools will in all probability remain a part of the Web, as researchers and users will continue to explore the boundaries of what hypermedia is and what it can be used to. An understanding of the strategies employed in Web augmentation is needed to make the next generation of Web augmentation tools easier to envision, develop and use.

We have investigated the recurring themes and techniques found in current Web augmentation tools. Based on this, we have developed and described the Arakne framework, and shown that the Arakne framework accommodates existing Web augmentation tasks, and that most tools can be modelled using the framework. A shared framework for these tools could benefit analysis and communication as well as development, and it is hoped that the Arakne framework is a step in the direction of the creation of such a framework.

The interesting and challenging part of creating a Web augmentation tool is not the nitty-gritty of interfacing to a Web browser or writing a proxy. The interesting part, at least in the author's experience, is to create tools that can help users structure their work and their browsing, and by implementing the Arakne applet some of the work needed to provide a full infrastructure for Web augmentation tools have been developed.

The Web has succeeded through (more or less) strict adherence to open standards that have been jointly developed by the interested parties. This has led to the development of standard tools usable for all. The continuation of this trend is crucial to the future development of the Web. The Open Hypermedia System initiative if supported by the hypermedia community can be become a shared standard from which the entire hypermedia community will benefit.

ACKNOWLEDGEMENTS

The author is a member of the Coconut project (http://www.cit.dk/coconut/), a joint research project consisting of Department of Computer Science, Aarhus University and Tele-Danmark Internet. The Danish National Centre for IT-Research (http://www.cit.dk/) supports the Coconut project.

The author wishes to thank Jesper Jühne for valuable discussion of Arakne, René Thomsen for adding to the code, Kaj Grønbæk for valuable advice, and the anonymous reviewers for good suggestions to structure this paper.

REFERENCES

[1] Anderson, K. M., Taylor, R. N., and Whitehead JR., E. J. (1994). Chimera: Hypertext for heterogeneous software environments. In *Proceedings of ECHT 94 Conference*, pp. 97–107, Edinburgh, Scotland.

[2] Anderson, K. M. (1997) Integrating Open Hypermedia Systems with the World Wide Web. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 157–166, Southampton, England.

[3] Bouvin, N. O. (1998). Designing Open Hypermedia Applets: Experiences and Prospects. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 281–282, Pittsburgh, USA.

[4] Bouvin, N. O., and Schade, R. (1999). Integrating temporal media with open hypermedia on the WWW. Submitted for publication.

[5] Bush, V. (1945). As we may think. In *The Atlantic Monthly* 176, 1 (July 1945), pp. 101–108.

[6] Carr, L. A., De Roure, D., Hall, W., and Hill, G. (1995). The distributed link service: A tool for publishers, authors and readers. In *Proceedings of the 4th International World Wide Web 95 Conference*, Boston, USA.

[7] Carr, L. A., Hill, G., De Roure, D., Hall, W., and Davis, H. (1996). Open information services. In *Computer Networks and ISDN Systems,* 28, pp 1027–1036, 1996.

[8] Carr, L. A., Hall, W., and Hitchcock, S. (1998). Link services or link agents? . In *Proceedings of the ACM Hypertext 98 Conference*, pp.113–122, Pittsburgh, USA.

[9] Davis, H., Hall, W., Heath, I., Hill, G., and Wilkins, R. (1992). Towards an integrated information environment with open hypermedia systems. In *Proceedings of the ACM Hypertext 92 Conference,* pp. 181–190, Milan, Italy.

[10] CritSuite. http://crit.org/http://crit.org/index.html

[11] Document Object Model. http://www.w3c.org/TR/REC-DOM-Level-1/

[12] Furuta, R., Shipman III, F. M., Marshall, C. C., Brenner, D., and Hsieh, H-W. (1997). Hypertext Paths and the World-Wide Web: Experiences with Walden's Paths. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 167–176, Southampton, England.

[13] Goland, Y., Whitehead, J., Faizi, A., Carter, S., Jensen, D. (1998). Extensions for Distributed Authoring on the World Wide Web – WebDAV. http://www.ics.uci.edu/~ejw/authoring/protocol/draft-ietf-webdav-protocol-08.pdf

[14] Grønbæk, K., and Trigg, R. H. (1994). Design issues for a Dexter-based hypermedia system. In *Communications of the ACM,* 37 (2) February, pp. 40–49, 1994.

[15] Grønbæk, K., and Trigg, R. H. (1996). Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. In *Proceedings of the ACM Hypertext 96 Conference*, pp.149–160, Washington DC, USA.

[16] Grønbæk, K., Bouvin, N. O., and Sloth, L. (1997). Designing Dexter-based hypermedia services for the World Wide Web. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 146–156, Southampton, England.

[17] Grønbæk, K., and Will, U. K. (1997). Towards a common reference architecture for open hypermedia. In *JoDI, Journal of Digital Information,* 1(2), 1997. http://jodi.ecs.soton.ac.uk/Articles/v01/i02/Gronbak/

[18] Grønbæk, K., Sloth, L., and Ørbæk, P. (1999). Webvise: Browser and proxy support for open hypermedia structuring mechanisms on the WWW. Submitted for publication.

[19] Guha, R. V., and Bray, T. (Eds.). Meta content framework using XML. http://www.textuality.com/mcf/NOTE-MCF-XML.html

[20] Hall, W., David, H., and Hutchings, G. (1996). *Rethinking Hypermedia: The Microcosm Approach.* Kluwer Academic Publishers, Norwell, USA.

[21] Hot Sauce. http://www.xspace.net/download/index.html

[22] HyperWave. http://www.hyperwave.com/

[23] Jühne, J., Jensen, A. T., and Grønbæk, K. (1998). Ariadne: A Java-based guided tour system for the World Wide Web. In *Proceedings of the 7th International World Wide Web 98 Conference*, Brisbane, Australia.

[24] Marshall, C. C., and Shipman III, F. M. (1995). Spatial hypertext: Designing for change. In *Communications of the ACM,* 38 (8) August, pp. 88–97, 1995.

[25] Marshall, C. C. (1998). Toward an ecology of hypertext annotation. . In *Proceedings of the ACM Hypertext 98 Conference*, pp. 40–49, Pittsburgh, USA.

[26] Maurer, H. (Ed.) (1996). *Hyper-G now, HyperWave: The next generation web solution*, Addison-Wesley, Harlow, 1996.

[27] Meyrowitz, N. (1989). The missing link: Why we're all doing hypertext wrong. In Barrett, E. (ed.). *The society of text: Hypertext, hypermedia and the social construction of information*, pp. 107–114, MIT Press, Cambridge, Massachusetts., USA, 1989.

[28] Microsoft Internet Explorer. http://www.microsoft.com/ie/

[29] Mozilla. http://www.mozilla.org/

[30] NCSA Mosaic. http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/

[31] Netscape Communicator. http://home.netscape.com/download/index.html?cp=djudepart

[32] Nürnberg, P. J., Leggett J. J., Schneider, E., and Schnase, J. L. (1996) Hypermedia operating systems: A new paradigm for computing. In *Proceedings of the ACM Hypertext 96 Conference*, pp.194–202, Washington DC, USA.

[33] The Open Hypermedia Systems Work Group. http://www.ohswg.org/

[34] Rittel, H. and Webber, M. (1973). Dilemmas in general theory of planning. In *Policy Sciences,* Vol. 4, 1973.

[35] Röscheisen, M., Mogensen, C., and Winograd, T. (1995) Beyond browsing: Shared comments, SOAPs, trails, and on-line communities. In *Proceedings of the 3rd International World Wide Web 95 Conference*, Darmstadt, Germany.

[36] Röscheisen, M., Winograd, T., and Paepcke, A. (1995). Content ratings and other third-party value-added information: Defining an enabling platform. In *D-Lib Magazine,* August 1995, http://www.dlib.org/dlib/august95/stanford/08roscheisen.html

[37] Shipman III, F. M., Furuta, R., Brenner, D., Chung, C-C., and Hsieh, H-W. (1998). Using paths in the classroom: Experiences and adaptations. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 267–276, Pittsburgh, USA.

[38] Shipman III, F. M., Furuta, R., and Marshall, C. C. (1997). Generating web-based presentations in spatial hypertext. In *Proceedings of the 1997 International Conference on International User Interfaces*, pp. 71–78, Orlando, USA.

[39] Stanley, T. (1998). Contextures: Focus + context + texture. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 295–296, Pittsburgh, USA.

[40] Trigg, R. H. and Weiser, M. (1986). TEXTNET: A network-based approach to text handling. In *ACM Trans. Office Information. Systems* 4, 1 (Jan 1986), pp 1–23.

[41] Trigg, R. H. (1988). Guided tours and tabletop: tools for communicating in a hypertext environment. In *ACM Trans. Office Information. Systems* 6, 4, pp 398–414, 1988.

[42] Web Squirrel. http://www.eastgate.com/squirrel/Welcome.html

[43] Whitehead Jr., E. J. (1997). An architectural model for application integration in open hypermedia environments. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 1–12, Southampton, England.

[44] Wiil, U. K. and Leggett, J. J. (1996). The HyperDisco approach to open hypermedia systems. In *Proceedings of the ACM Hypertext 96 Conference*, pp.140–148, Washington DC, USA.

[45] Wiil, U. K. and Leggett, J. J. (1997). HyperDisco: collaborative authoring and Internet distribution. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 13–23, Southampton, England.

[46] Wiil, U. K., and Østerbye, K. (1998). Using the Flag taxonomy to study hypermedia system interoperability. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 188–197, Pittsburgh, USA.

[47] XML – Extensible Markup Language. http://www.w3c.org/XML/

[48] Østerbye, K., and Wiil, U. K. (1996). The Flag taxonomy of open hypermedia systems. In *Proceedings of the ACM Hypertext 96 Conference*, pp.129–139, Washington DC, USA.

Integrating Temporal Media and Open Hypermedia on the World Wide Web

Niels Olof Bouvin & René Schade***

*Department of Computer Science,
University of Aarhus,
Aabogade 34A, DK8200 Aarhus N, Denmark

**Tele Danmark Internet
Olof Palmes Allé 36
DK8200 Aarhus N, Denmark

1 ABSTRACT

The World Wide Web has since its beginning provided linking to and from text documents encoded in HTML. The Web has evolved and most Web browsers now support a rich set of media types either by default or by the use of specialised content handlers, known as plug-ins. The limitations of the Web linking model are well known and they also extend into the realm of the other media types currently supported by Web browsers. This paper introduces the Mimicry system that allows authors and readers to link to and from temporal media (video and audio) on the Web. The system is integrated with the Arakne Environment, an open hypermedia integration aimed at Web augmentation. The links created are stored externally, allowing for links to and from resources not owned by the (link) author. Based on the experiences a critique is raised of the limited APIs supported by plug-ins.

2 KEYWORDS

Temporal media, open hypermedia, plug-ins, Web augmentation

3 INTRODUCTION

The World Wide Web has since its beginning steadily embraced more and more types of media. Today the average Web user will be exposed to pictures, video clips, sound recordings, music, and will interact with programs or 3D worlds residing on Web pages. These types of media are either handled by the Web browser itself or handled by specialised programs, 'viewers' or 'plug-ins'. Most media types are however supported/handled in the sense that it is possible to link to the entire media clip or to include it on a Web page, but not link from the media clip itself or to a segment of the media clip. Pictures are the exception, using image maps, to provide starting points for navigation. However all media types (HTML and otherwise) share the limitations of inline unidirectional links[1]; links cannot originate from documents not owned by the hopeful link creator, and the destinations of a link into a HTML document are limited to the named regions in the target document. These deficiencies are being addressed by integrating open hypermedia systems and the Web, allowing link structures to be stored externally of documents. This approach also allows for links to and from media types less amenable to modification than HTML, provided that suitable plug-ins or viewers are used.

This paper describes an open hypermedia integration to provide linking facilities to and from temporal media (such as video and audio clips). A search for an appropriate plug-in to provide the necessary functionality left the authors empty-handed. This resulted in the implementation of the Mimicry player which substitutes for a plug-in. Through the Mimicry controller, the system interacts with the Arakne Environment [4], allowing users to create multiheaded bi-directional links to and from temporal media clips, embedded or otherwise, as well as to and from HTML documents.

The results achieved by the Mimicry system (and the relative ease of implementing the ideas behind it) raise the question, why plug-in developers do not yet provide the functionality to support such a system. It would substantially ease the work of Web page designers, as media clips or parts thereof

[1] With the possible exception of image maps, which may have links defined externally.

could be reused, as well as supporting new (on the Web) technologies such as linking to and from temporal media.

The paper begins by introducing related work in the field of open hypermedia and on the Web. The merits of emerging standards such as SMIL, HTML+TIME, and XLink/XPointer are discussed. The Arakne Environment wherein the Mimicry system runs is introduced and described. The Mimicry system (player and controller) is described in detail and an example of Mimicry usage is given. Based on the experiences with the Mimicry system, the current state of plug-ins is discussed in the context of hypermedia systems. Finally directions for future work are discussed and a conclusion is reached.

4 RELATED WORK

This section will introduce the notion of externally stored link structures, open hypermedia systems, Web augmentation, and the work done with integration of temporal media and hypermedia systems.

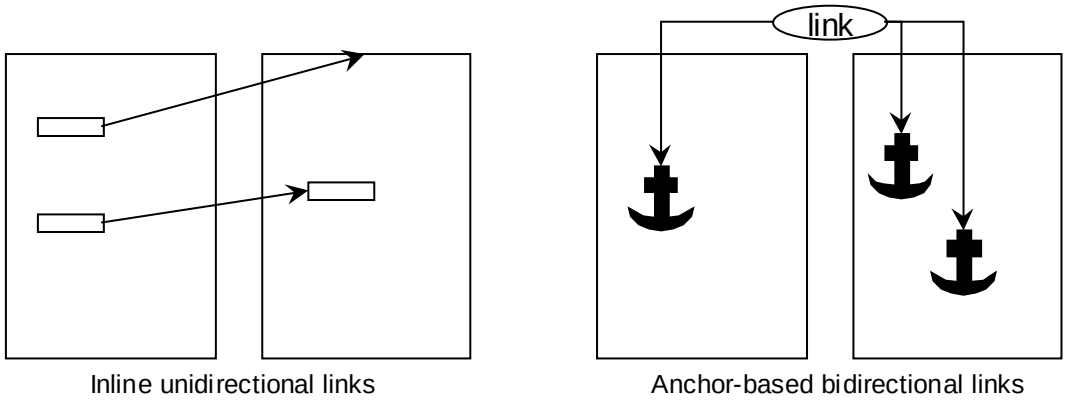


Figure 1 - Inline and anchor-based linking

4.1 Separating document and structure

The linking model found in the World Wide Web is based on inline unidirectional links. While simple and scaleable, this linking model is in some contexts inadequate in comparison with the linking model found in most modern hypermedia systems. Inline links are hard to maintain[2]; it is impossible to determine which links point to a page[3]; there can be only one set of links in a document; a link may point to only one destination rather than many; and this destination is limited to either a whole document or a named region therein. The problems with this approach can be illustrated by the following example: consider a situation, where a company decides on a Web based Intranet solution to allow easy access to their technical documentation. The Web, given the URL naming scheme, is very well suited for document distribution. The technical documents are crucial to several independent groups in the company and they all wish to put links into the technical document. Current Web technology yields two scenarios: either give each group access to copies modifiable by the respective group, or incorporate all links into one central document. The former makes updating the technical documentation difficult, and the latter will clutter the technical document with links interesting to one group, but irrelevant to the rest. Both cases leave the technical document open to undesirable modification. A safer and more maintainable solution would be to have one copy of the documentation available and then have each group use their own set of links into the documentation. This can not be done with the existing Web linking model.

---

[2] In the sense that links may point to documents that do no longer exist or have been moved.
[3] Save a brute force approach using search engines, which is computational expensive and not necessarily accurate.

A way to achieve this kind of flexibility is through the use of anchor-based hypermedia, which separates the anchor (or endpoint) and the link from the document. An anchor specifies a span in a document (up to the entire document), an area in a picture, a segment of a video clip and so forth. Links specify relationships between anchors, as seen in Figure 1. Anchors and links are stored outside the documents. A hypermedia application used by a user inserts links into documents *as they are retrieved* (e.g. not at the server, but at the client). The user can through the hypermedia application decide which sets of links to use, and the company described above would thus be able to maintain several sets of links to the same technical documentation.

There are pros and cons of this approach. As links and anchors are now separate they can be maintained separately and can be checked and updated independently of the documents they link. Links can be bi-directional, multiheaded and link into documents without modifying them. However, the insertion of links into the document on the fly carries some additional overhead, and does generally not scale as well as the simpler inline link model.

A great benefit of the anchor-based linking approach is that of opaque anchors, that is, the general system is not concerned with how an anchor addresses a selection in a media type. The general model need not be modified if a new anchor type is introduced to support a new media type, as long as the new anchor type adheres to the general anchor specification. The anchoring code (such as the ability to display an anchor into the media type) must of course still be written, but storage, link following, etc. is unaffected. This allows for complex anchoring constructs, and allows the developers to support new applications and media types without sacrificing existing work.

Anchors are created to match their media type. They must carry enough information to be able to identify the selection that the user had in mind when the anchor was created. In the case of text anchors, this information often consists of a selection and a context around the selection, so that it may be uniquely identified. An image anchor could (depending on its use) be as simple as two coordinate pairs to identify a rectangular selection, or be complex enough to identify arbitrary shapes as destinations. In the context of temporal media, it is natural to use time as unit in the system of coordinate. Frames are another possibility but the frame rate of a media clip may vary accordingly to the bandwidth of the user's connection. Furthermore some temporal media types, such as audio may not have frames at all.

4.2 Open Hypermedia and the Web

Open hypermedia systems are characterised by a focus on integration with third-party software. Historically, hypermedia systems have often been closed and monolithic, requiring other programs to comply to the standards of the hypermedia system in order to provide hypermedia functionality. This is problematic as it closes the door on existing non-compliant applications, and expects developers to change their programs – an unlikely proposition at best. Recognising that most people are not willing to throw away their applications in order to utilise a hypermedia system has led the open hypermedia community to integrate existing applications into their hypermedia systems instead. The level of which this is possible varies accordingly to the applications, ranging from the simple (show document) to the advanced (a full integration). Whitehead handles the implications of integrating third-party applications with open hypermedia integration in [21].

A natural consequence of the open hypermedia approach is the interest of integrating the Web into open hypermedia systems. Several groups in the field have created Web integration tools, and a non-exhaustive list includes DLS [5][6], DHM/WWW [8], Webvise [9], Navette [3], Chimera [1], and HyperWave [18].

A common approach to open hypermedia Web integration is to modify Web pages while en route to the Web browser. This modification usually consists of the addition of links or other kinds of structure. These links are stored on a structure server and are inserted into the pages using CGI-scripts, proxies, or programs controlling the Web browser. The interface presented to the user varies, ranging from no interface at all to full authoring applications allowing the users to modify and extend upon the existing collections of links and anchors. Most however allow the user to create links to and from whatever pages the user may desire, thus alleviating one of the major limitations of the existing Web architecture mentioned above. For a fuller discussion on the techniques of open hypermedia Web integration and Web augmentation in general, see [4].

The main target of these integrations has been Web pages rather than other kinds of Web-distributed media, as HTML is easily analysed and modified by use of proxies or other means. Other kinds of media that could be interesting include graphics and temporal media, streaming or otherwise. These data types are less readily modified, come in great variation, and requires viewers or plug-ins that may be difficult to integrate with an open hypermedia system.

4.3 Hypermedia and Temporal Media

Several hypermedia systems have been extended or devised specifically to handle temporal media. The most influential hypermedia model to incorporate the notion of temporal data is HyTime [7], which allows for multidimensional anchors (including the temporal dimension). HyTime is a general standard aimed at interchange and does as such not specify the interaction between hypermedia application and multimedia applications. While many systems have integrated temporal media one way or another, some systems go further, such as AEDI [2], which tries to ease work with large amounts of temporal data with structured indexing. Some systems try to facilitate automatic tracking and location of anchors in media clips; two well-known examples are Himotoki [10] and MAVIS [15]. While the possibility of selecting an object in one frame, and have the system automatically track the object in the rest of the video clip certainly is alluring, it is also quite computational expensive, and probably unlikely in a Web setting.

The ambitions of the authors are far more modest. We are merely interested in being able to create links to and from segments of temporal media. Future version may include anchors that cover an area of a video clip for some duration, but the area is not expected to move.

4.4 Temporal Media on the Web

The use of temporal media on the Web has steadily increased. It is today commonplace for news sites to bring either video clips or to provide access to whole TV shows online. Likewise the research community (notably the linguistic) has taken to making animations or sound recordings available. This opens for a hitherto unseen availability of valuable research material (such as historic film clips or language recordings), and therefore also an increasing demand to be able to interact with these media clips in new and innovative ways.

An example of an organisation beginning to put large amounts of temporal data on the Web is the Danish national library, Statsbiblioteket. In order to facilitate research, the library has made an increasing amount of historically interesting Danish sound files available on the Web [20].

4.5 Emerging standards

Some of the new emerging W3C standards are of special interest to the subject of this paper. This section will briefly describe these and discuss the implications for temporal media on the Web. It should be noted however that these are still evolving standards, and can be expected to undergo some transformation before being finalised.

SMIL [19] (Synchronised Multimedia Integration Language) is a recent Recommendation from W3C. It is an effort to support the layout and synchronisation of multimedia clips, e.g. synchronising a video clip with animations or a slide show with audio narrative and HTML documents. The standard is presentation oriented, and as such requires an authoring tool to modify.

HTML+TIME [12] (Timed Interactive Multimedia Extensions for HTML) is a proposed standard inspired by SMIL, and is concerned with the implementation of SMIL concepts in HTML. It thus adds a concept of timing to HTML and allows any HTML element to appear for a defined duration. Of special interest in this context is the timed hyperlink, and the integration of media players, which can be addressed and timed. Not only can the players be set to begin playing at a predetermined time, it is also possible to address inside the media clip and thus play a segment of a media clip, using the `clip-begin` and `clip-end` attributes. As HTML elements (in particular links) can be synchronised with other elements' timing events, it would be simple to create links that would synch with segments of a media clip, e.g. links appearing during a segment of a video clip and disappearing after the segment had been played.

As such HTML+TIME has certain requirements of content handlers that fit nicely with the requirements of links to and from temporal media. Specifically media players should be able to synch with events, as well allow the showing of segments. This is however a very new (proposed) standard and so far no Web browsers or media players the authors are aware of meet the HTML+TIME requirements.

A general problem with SMIL and HTML+TIME from the standpoint of the authors is that these standards address authoring of presentations and thus is in principle (though admittedly far more advanced) no different than the existing HTML authoring tools. The basic problem of links being created exclusively by the author of the document remains. These standards also address a somewhat different problem than linking to and from temporal media, as a key point of SMIL and HTML+TIME is synchronisation and presentation.

Both standards are, as noted above, still in development and may very well change in the future.

XML defines a way to describe structured data and documents. XPointer and XLink (XML Linking Language) are the accompanying resource location and linking standards. As of this writing, neither is in 'Recommended' state from W3C, but this is expected to happen sometime in 1999.

XPointer [17] is designed to specify locations in XML and other well-structured documents. The syntax is based on the Text Encoding Initiative 'extended pointer'. This is a rather compact and tree-oriented notation, which might take the following form: `id(foo).child(3,SEC).child(4,LIST)`. This expression would start at the entity identified as 'foo', continue at this entity's third child of the type 'SEC', and end with that child's fourth child of the type 'LIST'. This format is not XML, which might come as a surprise, but it has the advantage that it can be used in URLs.

XLink [16] is the language that ties XPointers together in links. XLink are not limited to XPointers – while endpoints in XML documents typically will be described with XPointers, XLink links can have any Web resource as a destination. Links may be in-line (as in HTML) or out-of-line, that is residing outside the linked documents. XLink supports bi-directional multiheaded links. Out-of-line links may be stored in simple text files or handled by link bases. XLink does not offer any protocol for such link servers.

XLink and XPointer are, from a Web augmentation standpoint, mainly interesting, if XML becomes widespread on the Web. While the linking constructs are quite powerful in XML documents, the

standards are not aimed at improving the state of the art with respect to HTML, that is not well-formed, nor does it address the intricacies of other media types, such as video or audio.

5 THE ARAKNE ENVIRONMENT

The Arakne Environment, shown in Figure 2, is a runtime environment aimed at supporting Web augmentation tools. The environment is primarily but not exclusively aimed at providing advanced hypermedia functionality to the Web. The environment is based on the Arakne framework [4], which is a general Web augmentation model, and was designed to be open and extensible. It currently supports a navigational hypermedia tool, Navette, and a guided tour tool, Ariadne [14].

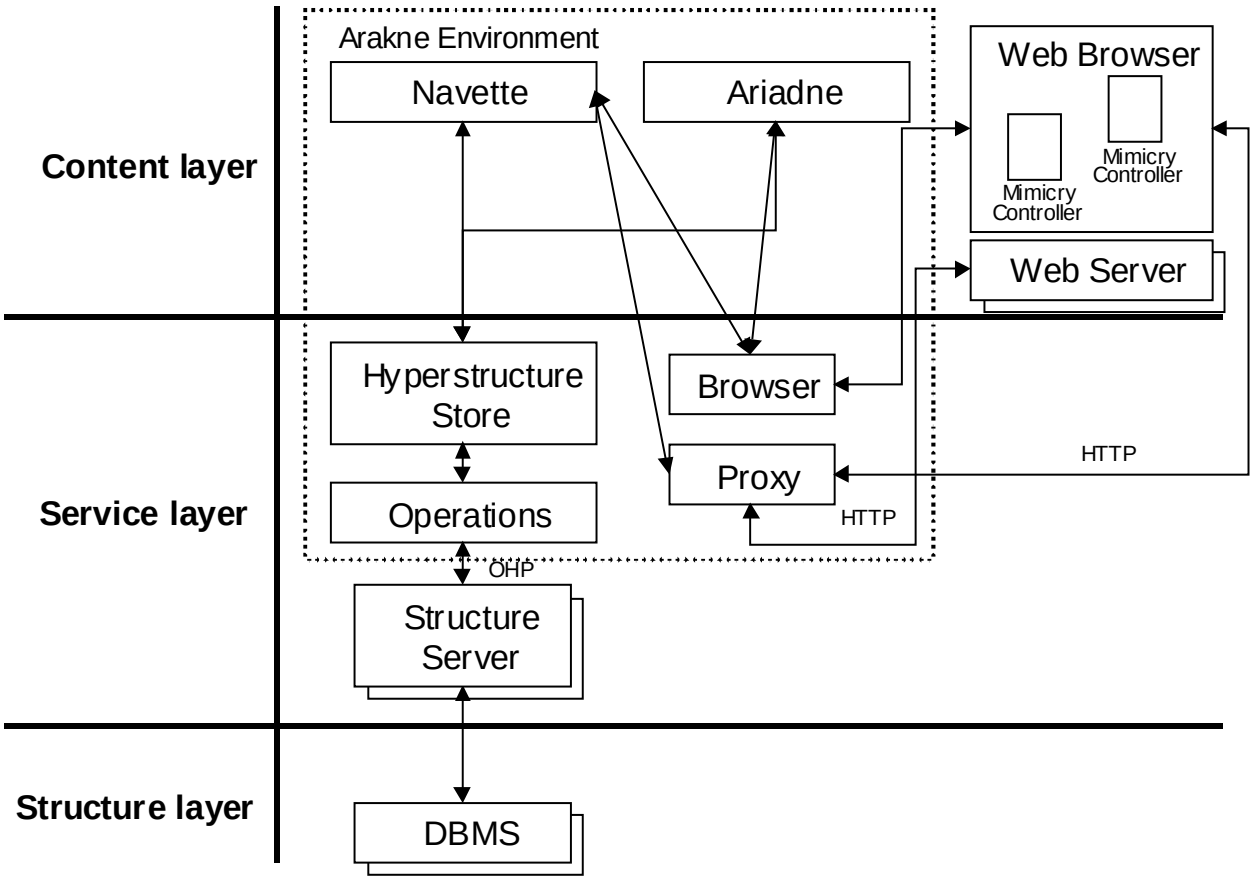


Figure 2 - The Arakne Environment

The framework may support any number of Web augmentation tools. These tools (known as 'navlets') are dependent on four core components of the Arakne framework: the Operations, the Hyperstructure Store, the Browser, and the Proxy. The navlet is the domain specific part of a Web augmentation tool. It provides a user interface as well as special logic to handle the specific domain. This may include deciding which links to display in a Web page based on information retrieved from the Hyperstructure Store component, or interfacing to the Proxy component for analysis of documents or to modify documents. Depending on the situation the computation and analysis may be carried out by the navlet or by another component.

The Operations component models the communication with the structure server layer. This component will thus typically support the same services as the structure server(s). This is where on the wire issues, such as network communication, marshalling, and multiplexing, are handled.

The Hyperstructure Store is the interface between the navlets and the Operations. The Hyperstructure Store provides convenience functions for the navlets, as well as caching the results of the queries retrieved with Operations. The Hyperstructure Store will also alert navlets to changes in the structures they subscribe to.

The Proxy component models the modification and analysis of Web content. Depending on their domain, navlets may require the Proxy to modify Web pages, and these requests for modifications are collected by the Proxy and used to modify the Web page. Other navlets may require access to the content of a Web page, which is also handled by the Proxy.

The Browser component models the user's Web browser. Through the Browser navlets can retrieve and modify the state of the Web browser such as which URL is currently displayed; the structure of the current frame set; whether a selection has been made in a frame and if so, what and where. Communication with plug-ins and applets running in the Web browser is also handled through the Browser component.

The situation depicted in Figure 2 is a situation of two navlets running in the Arakne Environment: Navette is a link creation tool, and thus needs access to the Proxy in order to insert links into Web pages. Ariadne is a guided tour tool and does not modify Web pages; and is thus not connected to the Proxy. Both however need to be able to tell and set the state of the currently displayed documents and to interact with the Web browser in other ways, as well as retrieving data from the structure server through the Hyperstructure Store.

By providing the components described above and by having an open architecture, the Arakne Environment aims to provide developers with an environment that allows for easy implementation of Web augmentation tools. The Arakne Environment is written in Java (but is not an applet), and currently integrates with the Microsoft Internet Explorer. The current version uses the DHMProxy [9] to insert links and other structures into Web pages. The relative ease of development has made the environment well suited for experiments such as the Mimicry system described herein.

6 THE MIMICRY PLAYER

The Coconut project has developed the Mimicry player to support linking to and from temporal media in the Arakne Environment. Realising that if we wanted to integrate temporal media into our hypermedia system, we would have to do it ourselves, as no existing plug-in seemed to offer an adequate API, we started looking for the easiest way to support temporal media.

The Java Media Framework [13] developed by JavaSoft, Intel, and Silicon Graphics is aimed at supporting temporal media in Java. The framework supports a wide range of video and audio formats[4], and more CODECs can be added. The Mimicry player is a Java Bean encapsulating the Java Media Framework player. It is basically interfaceless, but implements a rich API and event interface that can be utilised by other components.

The Mimicry controller is, referring back to Figure 2, an applet that communicates with the Navette tool through the Browser component; it thus acts as an intermediary between the Arakne Environment and the Mimicry player. The Mimicry controller provides the interface to the Mimicry player as well as to Navette. The Mimicry controller acts as the interface of the Mimicry player to the user. The controller has a control panel for the browsing and creation of anchors, which can be displayed by right clicking on the player window.

[4] Supported formats include AIFF, WAV, AVI, MIDI, MPEG-1, and QuickTime.

6.1 Web Page Modification by the DHMProxy

Multimedia files are normally presented in a Web browser using the appropriate plug-ins, according to the MIME type of the file. This is done either by using a direct link to the file or by embedding it into a Web page using <EMBED> or <OBJECT> tags. The Mimicry player is designed to mimic plug-ins, and will appear (to the user) as an ordinary plug-in on a Web page.

Rather than depending on Web page designers to adopt the Mimicry player as the standard viewer applet, the system utilises the DHMProxy [9] to modify Web pages, so that the Mimicry player is used instead of plug-ins.

The DHMProxy is aware of the formats supported by the Mimicry player and changes the Web pages accordingly. If a Web page embeds temporal media using <EMBED> or <OBJECT> tags, these tags are replaced with a corresponding <APPLET> tag with the same layout dimensions. If a Web page has a direct link to a temporal media file, the DHMProxy returns a Web page containing the <APPLET> tag in the body of the page. Using this approach it is possible to translate plug-in invocations into applet invocations, without changing the layout of the Web pages.

The DHMProxy also takes care of inserting anchors from the chosen link collections on the structure server into the Mimicry controllers. All the anchors residing in a media clip are passed on to the applet in parameter tags consisting of name, id, and time span. When the Web page is loaded in the Web browser and the Mimicry controller applet is launched, it is thus aware of the anchors in the media clip. This modification or decoration is basically similar to the ordinary text decoration (that is, insertion of links) done by the DHMProxy.

6.2 Mimicry in Action

In order to present Mimicry in action, we have started the Arakne Environment, which launches the Web browser[5], as presented in Figure 3. The Web browser has been configured to use the DHMProxy. As we want to create links into Web pages and media files, we have started Navette in the Arakne Environment.

The situation in Figure 3 is as following: another user has earlier created a link with the name ‘Link 10’ in the link collection ‘Hyperspace1’, which contains three anchors. The first anchor ‘Harrison Ford’ originates in a Web page containing a description of the actor Harrison Ford. The second anchor ‘Quote’ originates in a Web page containing famous movie quotes, referring a specific quote. The third anchor ‘Endpoint 14’ originates in a movie clip embedded in a Web page, referring to a 59 seconds long segment of the 2:02 minutes long clip, that features the quote.

Browsing the Web using the Arakne Environment, we encounter the Web page containing the movie quotes. Since we have ‘Hyperspace1’ opened, the DHMProxy has decorated the page with the ‘Quote’ anchor. The anchor is presented, so that we can follow ‘Link 10’ either to the media segment ‘Endpoint 14’ or to the Web page containing the anchor ‘Harrison Ford’. We decide to follow the link to the media anchor and the Web page containing the video clip is loaded, as shown in Figure 3.

The DHMProxy decorates the Web page by substituting the plug-in tag with an applet tag and anchors from the structure server. When the Mimicry controller applet is launched it retrieves its anchors from the parameter tags, alerts the Arakne Environment to its existence and asks the Mimicry player to start downloading the media file. In Navette ‘Link 10’ is opened and the ‘Endpoint 14’ anchor is in focus.

[5] The current version is integrated with the Microsoft Internet Explorer

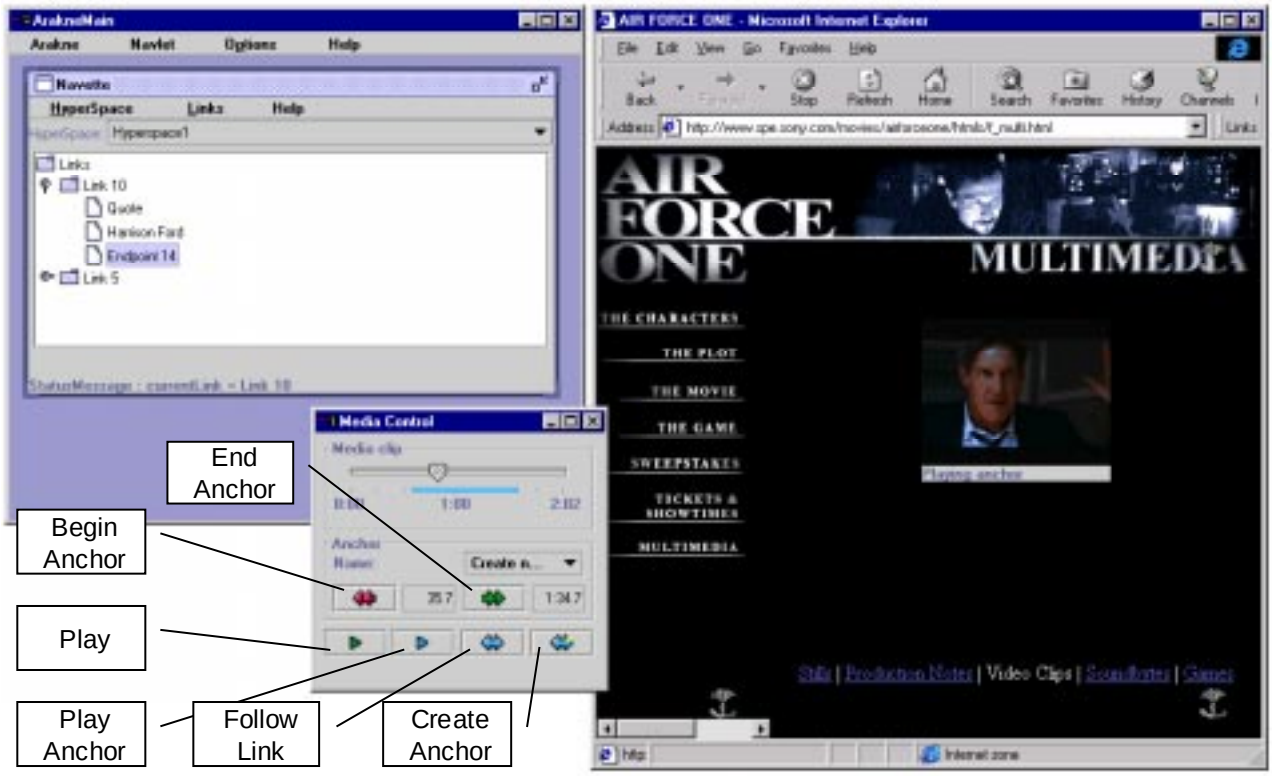


Figure 3 - Playing the anchor endpoint "Endpoint 14"

Clicking on the Mimicry control applet, the 59 seconds segment ‘Endpoint 14’ will be played, starting at 25.7 seconds and ending at 1:34.7. Interested in the movie clip we right-click on the applet and a popup menu appear. In the popup menu it is possible to select other anchors in the clip or to open the control panel. We choose “open control panel”, and the control panel, on top of the Web browser and the Arakne Environment in Figure 3, is presented. The control panel consists of a slider, indicating the length and current position of the media clip, buttons for creating/editing anchors and a dropdown menu containing all the anchors in the media clip. The current anchor is drawn on the slider.

Dragging the slider to the start position and pressing the “Play” button causes the Mimicry player to start playing from the beginning. While watching the media clip, we notice the actor Gary Oldman, and decide to find more information about him. In the dropdown menu containing all the anchors in the media clip, we find another media anchor named ‘Endpoint 11’. Selecting ‘Endpoint 11’ and playing it, we realise the anchor is covering the part of the clip depicting Gary Oldman. Wondering if the link is about the actor, we click the “Follow Link” button on the control panel. The browser loads another URL, which is a direct link to another movie clip, resulting in the situation shown in Figure 4.

Another Mimicry controller is launched and Navette has changed its focus to ‘Endpoint 12’ in ‘Link 5’. Clicking the Mimicry controller, the segment ranging from 1:04.8 to 1:33.7 of the movie file is played. The ‘Endpoint 12’ segment does indeed refer to Gary Oldman appearing in another movie trailer.

Watching the movie trailer from the beginning, we find yet another actor, Matt LeBlanc. Since we have had the pleasure of following links created by others, we would like to add an additional link to the current link collection. We know where to find further information about Matt LeBlanc and de-

cide to create this relation. In Navette we deselect the current link. In the control panel, we create a new media anchor using the “Begin Anchor” and “End Anchor” buttons, editing a few times, reviewing the new anchor several times with “Play Anchor” and finally end up with exactly the portion of the clip concerning Matt LeBlanc. We press “Create Anchor” in the control panel, and as no links are selected in Navette, a new link ‘Link 25’ is created, initially containing the anchor ‘Endpoint 15’. The Arakne Environment stores the new link and anchor on the structure server. We browse the Web to a Web page containing more information about the actor as shown in Figure 5.

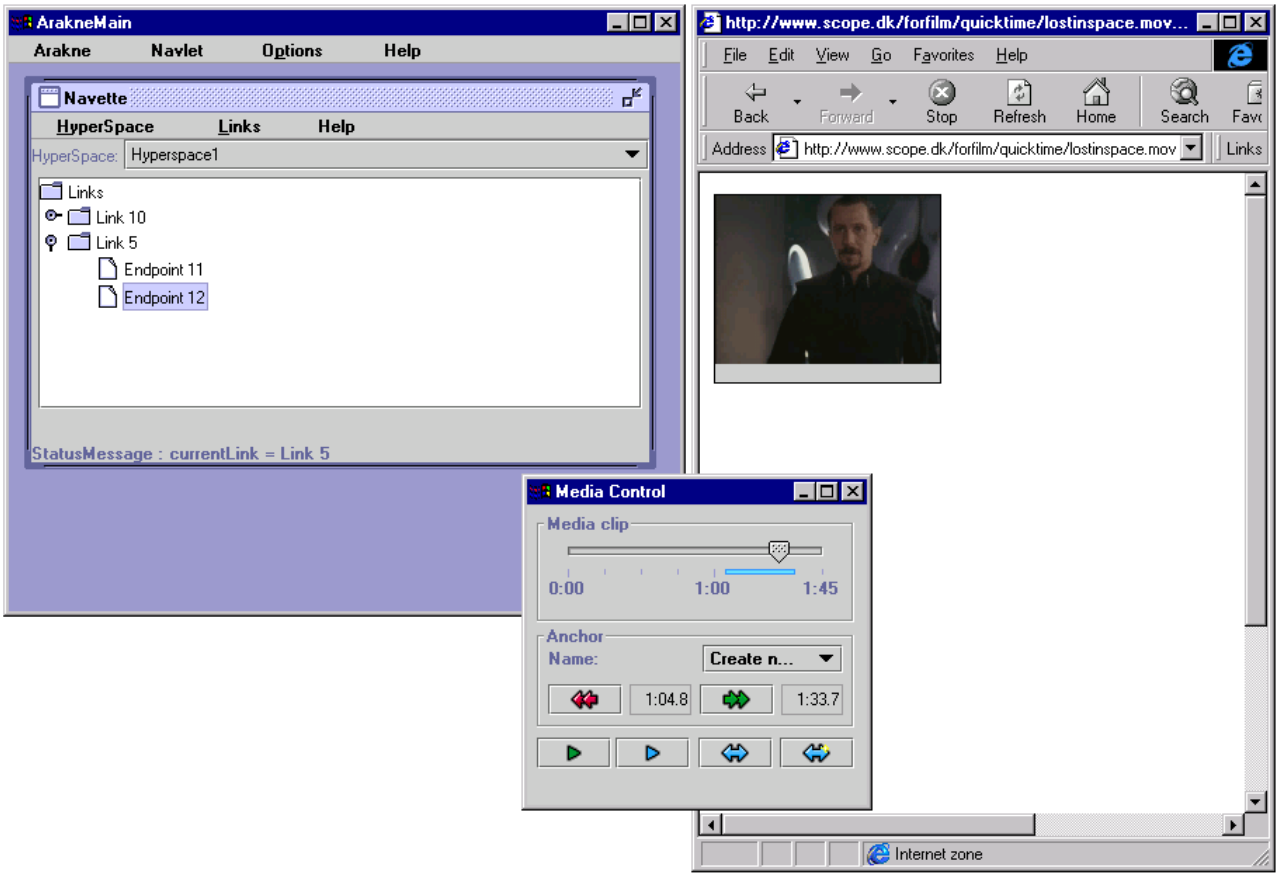


Figure 4 – Playing the anchor ‘Endpoint 12’

When the Web page is retrieved, we highlight the text ‘Matt LeBlanc’ in the Web browser, right click and select “Add Anchor” in the popup menu[6]. This information is sent to Navette, which reacts by creating the endpoint “Matt LeBlanc”. Pressing the refresh button in the Web browser forces the DHMProxy to decorate the Web page with the newly created anchor. In the Web browser, the new anchor is presented as a link in ‘[*]’ next to the Matt LeBlanc text. To test the link, we click it, and the media clip from Figure 4 is loaded, ready to play ‘Endpoint 15’. If we open the control panel and press the “Follow Link” button the Web page from Figure 4 is loaded. We have now created a link in the link collection ‘Hyperspace1’. Next time another user is browsing the Web using the Arakne Environment and the ‘Hyperspace1’ link collection, he or she will be able to see and follow these links.

[6] The essential commands for link creation with Navette are available through the use of right-click menus on highlighted text in the Microsoft Internet Explorer.

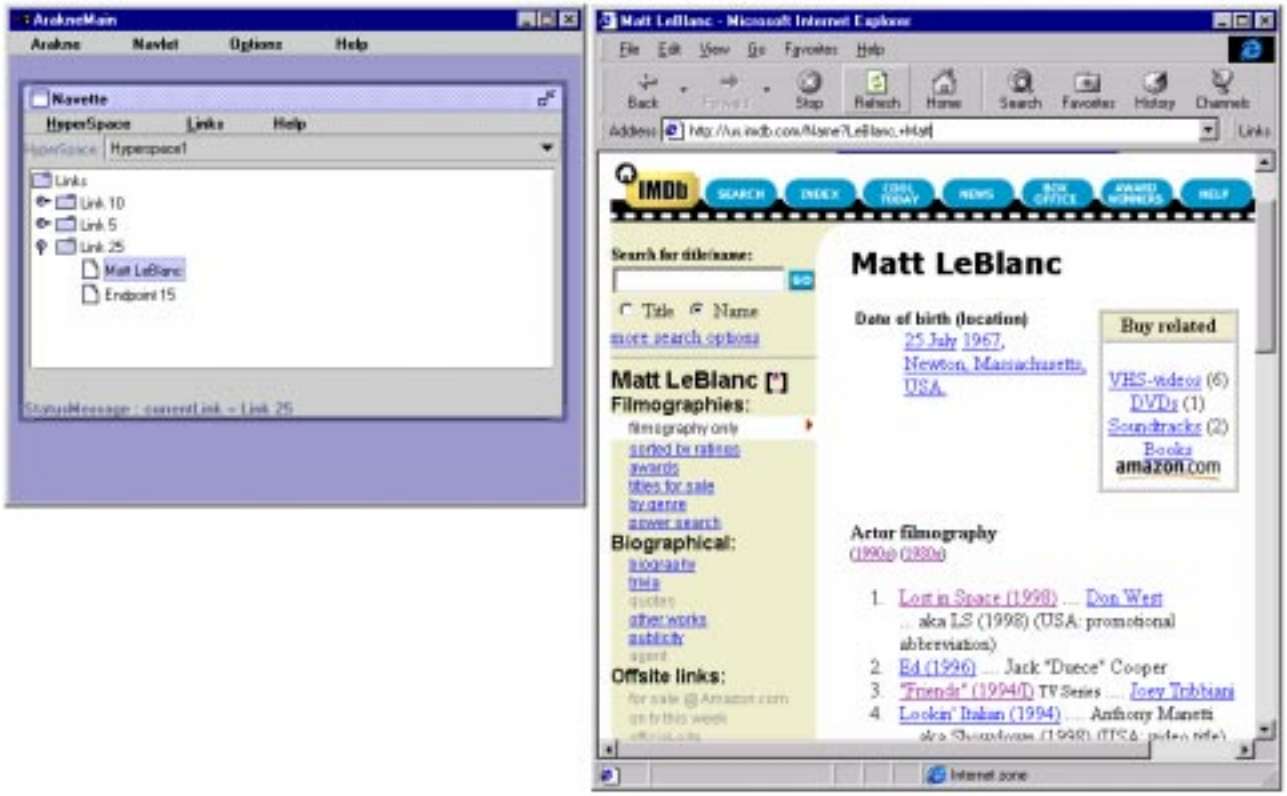


Figure 5 – Creating a HTML anchor using Navette

We have now described how to follow a link to a media clip using Mimicry, how to follow links between two media clips, and how to create anchors and links. In Figure 3 the Mimicry is used as embedded on an HTML page, and in Figure 4 Mimicry is used standalone as a direct link to a media clip.

7 DISCUSSION

The following will discuss the implication of the results in the context of plug-in developers.

7.1 The Problem with Current Plug-ins

Plug-ins are used to handle media types not supported natively by the Web browser. Plug-ins can be controlled at runtime through LiveConnect [11] using JavaScript. In order to support linking in and out of temporal media, we need to continuously be able to get and set the state of the plug-in. The degree of openness to this kind of control varies tremendously. The most extreme example of an open plug-in handling temporal media that the authors have been able to locate is the Beatnik plug-in published by Headspace. Beatnik is a plug-in targeted at sound and music files, and has a very rich API. The APIs supported by some popular plug-ins are shown in Table 1. Beatnik is sound only, and is as such only used as a comparison to the other media players, as well as Microsoft Media Player which cannot be controlled through JavaScript.

Apart from Beatnik, no plug-in allows for the playing of a designated but not in the media predefined segment. This ability is fundamental to link following in temporal media, and it is thus not possible to create a hypermedia system supporting links to and from media clips with these plug-ins. The rich authoring environments of some media tools, especially QuickTime[7], make the lack of support for modifications at run-time more grating. The author should certainly have a very rich authoring tool

[7] However, QuickTime offers another approach as mentioned in section 8.

at his or her disposal, but that does not eliminate the need to be able to dynamically alter the playback of a media clip, as it is shown on a Web page.

Plug-in/Content Handler	**Methods**[1]	**Callback Methods**[1]
Real RealPlayer	SetSource DoPlayPause DoNextItem[2] DoPrevItem[2]	onClipOpened onGoToURL
Apple QuickTime	[3]	[3]
HeadSpace Beatnik	getPlayLength getPosition setPosition setStartTime setEndTime	onLoad onReady onStop
Microsoft Media Player[4]	GetCurrentPosition SetCurrentPosition GetSelectionStart SetSelectionStart GetSelectionEnd SetSelectionEnd	

[1] This is not a comprehensive list, but merely methods relevant to linking. [2] These methods require that the items are defined by the author of the media clip. [3] QuickTime does only support arguments to the `<embed>` tag. [4] The methods described are through the COM-interface, not JavaScript.

Table 1 - Supported APIs of various plug-ins

Integration with plug-ins requires some degree of openness on part of the plug-in API. In order to support linking (and other kinds of integration) it must at least be possible to designate a segment of a recording and to allow that segment to be played. Specifically we would suggest that the methods outlined in Table 2 could the basic API of any plug-ins handling temporal media.

Methods	Callback Methods
getSourceURL	onLoad
setBeginClip	callbackWhen
getBeginClip	
setEndClip	
getEndClip	
getDuration	
getCurrentTime	
playPause	
playClip	

Table 2 - Basic API requirements for temporal media plug-ins

A plug-in would thus be able to play a designated clip or segment of a media clip and would report its progress through the media clip. This API would allow a developer to support the kinds of interactions supported by the Mimicry player through the use of JavaScript and LiveConnect [11]. This API is simple and should be easy to implement for plug-in developers. The Beatnik plug-in clearly demonstrates that this is indeed achievable.

Another solution investigated by the Coconut project is to make a specific integration with a media player, in this case the Microsoft Media Player that through its COM-interface supports a rich API. External applications would register events from the Media Player and through the COM-interface control it. Being a COM-component the Media Player can be integrated into a user interface supporting anchor and link creation. This approach does have some limitations. It is platform dependent, and unless a proxy is used to modify the Web pages as described below, the supporting software is

forced to change content handler while the Web page containing the media clip is being displayed. This is possible, but it is neither elegant nor seamless.

8 FUTURE DEVELOPMENT

Unless a media player emerges that satisfies the needs of temporal media linking, the authors will continue to work on the Mimicry player. Some current development holds promise with regards to future versions. Future versions of the Java Media Framework will support more media formats and sport additional improvements. An interesting development would be Java Media Framework support for streaming, which is currently offered by Real Networks, but this version is so far limited to Netscape Communicator.

As media players and Web browsers supporting SMIL and HTML+TIME become available, we will investigate the possibilities of linking into the new structures supported by these standards, as well as using the new generation of plug-ins and viewers supporting these formats.

Apple has created QuickTime for Java, featuring an API similar to the Java Media Framework, being able to playback formats supported by the QuickTime plug-in 2.0 in an applet. This may be an area worthy of future investigation.

The current Mimicry player is a Java applet, and a clear future direction would be to convert the player into a plug-in. This would probably eliminate much of the need for the intervention by DHMProxy, as the Mimicry plug-in would automatically launch on its registered MIME types.

9 CONCLUSION

We have described a system allowing users to dynamically create links to and from temporal media on the Web regardless of the users' ownership of the Web pages or media clips involved. The system has not been created to compete with existing media players, but merely to allow for experimentation and to highlight the lack of support for dynamic uses of temporal media currently found in most plug-ins. As such it has been successful.

The future of temporal media on the Web is a bright one. Bandwidth will continue to rise and emerging standards such as SMIL and HTML+TIME will support the use of temporal media in new ways. Whether these standards will converge to support only presentation, or open for more dynamic uses remains to be seen.

10 ACKNOWLEDGEMENT

The authors are members of the Coconut project (http://www.cit.dk/coconut/), a joint research project consisting of Department of Computer Science, Aarhus University and Tele Danmark Internet. The Coconut project is supported by the Danish National Centre for IT-Research (http://www.cit.dk/).

The authors wish to thank Niels Husted and René Thomsen for adding to the code, Peter Ørbæk for creating the DHMProxy, and the anonymous reviewers for good suggestions.

11 REFERENCES

[1] Anderson, K. M. (1997). Integrating Open Hypermedia Systems with the World Wide Web. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 157–166, Southampton, England.

[2] Auffret, G., Carrive, J., Chevet, O., Dechilly, T., Ronfard, R., and Bachimont, B. (1999). Audio-visual-based hypermedia authoring: using structured representations for the efficient manipulation, of AV documents. In *Proceedings of the ACM Hypertext 99 Conference*, pp. 169–178, Darmstadt, Germany.

[3] Bouvin, N. O. (1998). Designing Open Hypermedia Applets: Experiences and Prospects. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 281–282, Pittsburgh, USA.

[4] Bouvin, N. O. (1999). Unifying Strategies of Web Augmentation. In *Proceedings of the ACM Hypertext 99 Conference*, pp. 91–100, Darmstadt, Germany.

[5] Carr, L. A., De Roure, D., Hall, W., and Hill, G. (1995). The distributed link service: A tool for publishers, authors and readers. In *Proceedings of the 4th International World Wide Web 95 Conference*, Boston, USA.

[6] Carr, L. A., Hall, W., and Hitchcock, S. (1998). Link services or link agents? . In *Proceedings of the ACM Hypertext 98 Conference*, pp.113–122, Pittsburgh, USA.

[7] DeRose, S. J., and Durand, D. G. (1994). Making HyTime Work. Kluwer Academic Publishers, 1994.

[8] Grønbæk, K., Bouvin, N. O., and Sloth, L. (1997). Designing Dexter-based hypermedia services for the World Wide Web. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 146–156, Southampton, England.

[9] Grønbæk, K., Sloth, L., and Ørbæk, P. (1999). Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on the WWW. In *Proceedings of the 8th International Conference on the World Wide Web*, Toronto, Canada.

[10]Hirata, K., Hara, Y., Takano, H., and Kawasaki, S. (1996). Content-oriented Integration in Hypermedia Systems. In *Proceedings of the ACM Hypertext 96 Conference*, pp. 11–21, Washington D.C., USA.

[11]Hoque, R. Java, JavaScript and Plug-In Interaction Using Client-Side LiveConnect. http://developer.netscape.com/docs/technote/javascript/liveconnect/liveconnect_rh.html

[12]HTML+TIME. http://www.w3.org/TR/NOTE-HTMLplusTIME

[13]Java Media Framework. http://www.javasoft.com/products/java-media/jmf/index.html

[14]Jühne, J., Jensen, A. T., and Grønbæk, K. (1998). Ariadne: A Java-based guided tour system for the World Wide Web. In *Proceedings of the 7th International World Wide Web 98 Conference*, Brisbane, Australia.

[15]Lewis, P. H., Davis, H. C., Griffiths, S. R., Hall, W., and Wilkins, R. J. (1996). Media-based Navigation with Generic Links. In *Proceedings of the ACM Hypertext 96 Conference*, pp 215–223, Washington D.C., USA.

[16]Maler, E., and DeRose, S.J. (Eds.). (1998). *XML Linking Language (XLink) Design Principles*. http://www.w3.org/TR/NOTE-xlink-principles

[17]Maler, E., and DeRose, S.J. (Eds.). (1998). *XML Pointer Language (XPointer)*. http://www.w3.org/TR/WD-xptr

[18]Maurer, H. (Ed.) (1996). *Hyper-G now, HyperWave: The next generation Web solution*, Addison-Wesley, Harlow, 1996.

[19]SMIL. http://www.w3.org/AudioVideo/

[20]Statsbiblioteket. Dansk Lydhistorie. http://www.sb.aau.dk/dlh/

[21]Whitehead Jr., E. J. (1997). An architectural model for application integration in open hypermedia environments. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 1–12, Southampton, England.

12 VITAE

Niels Olof Bouvin is a Ph.D. student in Computer Science at University of Aarhus, Denmark. His research interests include open hypermedia systems, Web augmentation, structural computing, and collaboration on the Web. He is currently involved in the Coconut project, a co-project between the Department of Computer Science and Tele Danmark Internet. Niels Olof Bouvin received his master's degree in 1996 from Department of Computer Science, University of Aarhus, Denmark.

René Schade is a system developer at Tele Danmark Internet, Denmark. He is currently working at the Coconut project, a co-project with the Department of Computer Science, University of Aarhus, Denmark. He finished his master degree in 1997 from the Department of Computer Science, University of Aarhus. His research interests are: World Wide Web; Hypermedia and Multimedia and Dynamic Programming Environments.

Experiences with OHP and Issues for the future

Niels Olof Bouvin

Department of Computer Science, University of Aarhus,
Aabogade 34, 8200 Århus N, Denmark
n.o.bouvin@daimi.aau.dk

Abstract. The OHSWG has by now moved from specifications to running code. This is an important step, not only because this is the only way of maturing the specifications, but also because it strengthens the credibility of the OHSWG. Showing that the ideas expressed by the OHSWG can be implemented is however not enough, at least not if we desire a wider audience than ourselves. Concomitantly the XLink standard has begun to take shape, metadata on the Web seems to be on the rise, and interconnectedness is sharply rising with various small devices (such as WAP phones and PDAs) gaining access to the Internet. We are living in interesting times. Based on the experiences of developing the Arakne Environment, the author attempts to point out some worthwhile directions for future work within the OHSWG.

1 Introduction

The original goal of the Open Hypermedia Systems Working Group was to identify what constituted open hypermedia and to devise a common standard, so that open hypermedia systems could interoperate, and possibly share access to integrated third-party applications. Some of these goals have now been met. There is a common standard, and the Group has now several times at Hypertext conferences demonstrated heterogeneous hypermedia systems working together through the Open Hypermedia Protocol (OHP).

Existence has thus been established, and yet we do not find the world breaking our door down in order to get hold of this amazing technology. The author has over the last few years designed and developed an open component-based hypermedia system, the Arakne Environment, which is now based on the OHSWG compliant Construct servers. The experiences with converting to OHP are related in this position paper with some pointers for future concern are raised, especially with regards to future standardization work.

2 The Arakne Environment

The Arakne Environment has been described in ([2], [4], [3]), and for the purposes of this paper there is no need to go into great detail. Arakne is a framework for open hypermedia programs, and currently supports navigational, temporal, guided tour, and spatial hypermedia on the Web. The system has gone through

several iterations and now uses the OHSWG compliant Construct servers as structure servers. The original design idea was to create a system where functionality could be easily added without affecting the rest of the system — a hypermedia plug and play system. Over the design and development iterations this ideal has come closer.

3 Experiences with the development of the Arakne Environment

This section will describe the process of converting the Arakne Environment to a native OHP application. As Arakne started out as a Ph.D. project there was never existing legacy applications, allowing for radical redesigns (such as replacing the hypermedia protocol used in the original version with OHP).

3.1 The transition to OHP

The Arakne Environment did not start out using OHP. The original version relied on the Devise Hypermedia servers, but as the Construct servers were completed within the same project as Arakne, the transition to OHP began. One advantage in the context of this conversion was the code sharing between server and client, as both are written in Java. As OHP is language independent, the transition would under all circumstances have been possible albeit admittedly more time consuming. The open source nature of Construct does not limit the advantage of code reuse to the developers associated with Construct.

The first version of Arakne to utilize OHP was version 2.0. This was not a complete transition, as only the components that communicated with the structure servers were changed — the rest were (virtually) unchanged. This was a relatively simple operation as the original data model was based on DHM [12] (and thus Dexter [13]), and the gap to the OHSWG data model was relatively minor. The components communicating with the Construct servers would then transparently convert between the Arakne and the OHSWG data models, e.g. folding OHSWG anchors and endpoints into Arakne endpoints and vice versa. This 'hack' allowed us to get up to speed relatively rapidly, so that we could commence testing of the structure servers forthwith.

Given the nature of hypermedia conversion [15], this first step did not use the full functionality of the Construct servers. The features not supported by the hypermedia applications (known as 'NavLets' or 'views') could naturally not be utilized on the structure servers. The most important missing feature was the support for sessions, which is a key component in the OHSWG support for collaborative work.

This, and extensibility concerns, led to a complete redesign of the Arakne architecture and a full adoption of the OHSWG data model, resulting in the current version 2.1. This was a considerably more involved task as all existing components in the framework were either modified or completely rewritten and new components handling sessions and collaboration had to be added.

The current architecture allows users to be engaged in an arbitrary number of sessions using any number of hypermedia views. Each session has its own data model, coupling mode, and views, independently of other sessions. This has been accomplished without major changes in the views, while still allowing them room to extend support for collaboration between users as the view developers see fit.

The author would like to see more hypermedia systems support OHP, and there are some lessons that can be learned from the experience of converting Arakne to OHP.

The first lesson is that the system actually works! OHP can be used for more than once a year Hypertext Conference demonstrations. Arakne along with the Construct servers are thus among the growing number of validations of the basic design of the OHP. One of the original goals of OHP was to allow dissimilar hypermedia systems to interoperate and to share third party application integrations. In this context the Arakne Environment is probably special, as it does no longer have a hypermedia protocol of its own — it has fully embraced OHP.

The second lesson is it is not necessary to natively[1] support OHP. Using an extra component to convert from one data model to another is close in principle to the shim of [6], though the conversion is from a protocol to an API rather than from protocol to protocol. It should be noted that the task of converting Arakne to OHP was considerably smaller than e.g. establishing interoperability between Chimera and HyperDisco [19], where the target was true interoperability (we just wished to move Arakne from DHM to OHP), and where there were significant dissimilarities between the systems, including features, programming language, and protocol format. DHM and OHP are fairly close — both utilize XML (or at least tagged ASCII) for transport, the data model are similar, and the transition to OHP was made easier by the existence of available Construct Java libraries. The method used to integrate Chimera and OHP (under development as this is written) is in principle similar to the one used by the version 2.0 Arakne. Existing Construct libraries are plugged into Chimera with additional code to handle the conversion back and forth. This is a more ambitious and exciting integration, as Chimera (as opposed to Arakne 2.0) natively supports sessions.

3.2 Adding new functionality — "Embrace and Extend"

One of the original design goals of the Arakne framework was to provide a general framework for all Web augmentation hypermedia tools. In practice the first version integrated Navette (a navigational hypermedia tool and a successor to DHM/WWW [9]) and Ariadne [14] (a guided tour tool not originally developed for Arakne). The latest hypermedia tool to be integrated with Arakne is CAOS [17] (a spatial hypermedia tool), which also originally was not developed for Arakne. By having an open architecture, Arakne has thus been able to "embrace and extend" already existing hypermedia tools. As support for collaborative work is now an integrated part of Arakne, this functionality is made available for the new views.

[1] i.e. abandon the existing data model etc. in lieu of OHP

3.3 New protocols

New structural domains (such as spatial hypermedia [16]) are emerging and are swiftly becoming one of the most dynamic and interesting areas in hypermedia (as witnessed by popularity of the Structural Computing workshops). Hypermedia is no longer "just" navigational links and anchors.

OHP handles navigation and collaboration through sessions. The Construct servers have been extended to provide additional services through the OHP-Comp (composites) and OHP-Space (spatial hypermedia). New services can be added to the Construct servers and Arakne, as they become available. Indeed, with the development of the Esbjerg CSC tools [18], adding new services has become even easier.

The new structural domains are areas where the OHSWG can contribute tremendously to the hypermedia community at large by providing solid protocols and structure servers. We have the knowhow and the robust servers, and should be able to help other workers in the hypermedia community. The active discussions following the presentation of CSC by Wiil and Nürnberg at HT2000 were witness to that this is an idea whose time has come.

3.4 The evolution of OHP

It should be noted that the OHP of the Construct servers does not fully follow the Darmstadt DTD. This is not so much of a problem as it might first seem, for standards must necessarily be validated through implementation. Some changes and clarifications became necessary and were subsequently implemented. It should however be clear that the changes made to OHP must be documented and published, in order to support further discussion and development of the OHP. The issue of proper protocol documentation will be dealt with in section 4.

4 The need for further standardization

A basic requirement of any standard is a precise and unambiguous definition. Unfortunately, the current OHSWG OHP specification is lacking in this respect. This leads to problems with interoperability between different implementations, exactly what OHSWG was supposed to solve! This section will describe the problem and propose possible solutions.

4.1 The problem with XML

One result of the OHSWG 4.5 meeting was creation of the On the Wire group, who was given the task to determine what transport layer the Open Hypermedia Protocol should utilize. The author was a member of this group, and the eventual recommendation was to use XML documents over sockets as the basic entry point, with CORBA support left for more advanced servers.

The rationale behind choosing XML was fairly solid: XML is easy to parse with free XML parsers available for most programming languages; it is human

readable which eases debugging considerably; and using validating XML parser guarantees adherence to a given grammar. All this combines to make the entry into OHP relative easy and painless. Furthermore one should not be blind to necessity of being "buzzword-compliant" in this wired world, and XML certainly fulfills that purpose.

One problem that has emerged over time with XML is the level of detail expressible with the DTD grammar. DTD is well suited for general syntactic declarations, but cannot express semantics nor more fine-grained syntactic aspects such as data types or ranges. This is witnessed by the proliferation of `#CDATA` (i.e. unspecified character data) in most DTDs. Thus the specification is not enough by itself. It must be accompanied by either exhaustive documentation or code implementing the standard, if not the implementation will be left to guesswork by the developer. Data and code are closely integrated in this setting, and this has several disadvantages. XML was originally set out (though admittedly like much Internet technology, it was over hyped at the time of its introduction) to ease data interchange by creating solid specifications to separate code and data. The reality is that while this goal has been partly accomplished, the devil (as always) is in the details and this is where XML is still lacking.

The lack of detailed specification has also hurt OHP. The Darmstadt DTD that forms the (excellent) basis for OHP contains many instances of `#CDATA` where the details are left to the implementation. While this is just a set of decisions to be made when dealing with one implementation of OHP (such as Construct), it becomes a problem when interoperability — one of the purposes of OHSWG — is the goal. As mentioned in section 3.4, the development of Construct led to a number of choices with regards to what OHP is. Another implementor of OHP can however not be relied on making the same choices — the hectic work leading up to last year's OHSWG demonstration showed the problems with this approach, as there were several areas where the different implementations had to be modified to accommodate the differences. One solution to this is to adopt an existing implementation; this is the case of the current OHP/Chimera integration (mentioned in section 3.1), but this requires a certain compatibility between the systems with regards to e.g. programming languages.

It was suggested by Peter J. Nürnberg at OHSWG 5.5 that the time has come for OHSWG to adopt a reference OHP implementation. The author agrees with this notion, but would like to see it extended. One of the requirements of a proposed standard submitted to the W3C is that there should be a clear specification and at least two independent and compatible implementations of the specification. This is a very important point, as it lessens the probability of idiosyncrasies in one implementation determining the specification and for tacit, unspecified details to make it into the final standard. If we are to move forward to make OHP a "proper" standard (at standard bodies such as IETF or W3C), as discussed in 5, we *must* address this issue. As long as we maintain compatibility OHSWG can only benefit from such a move. One benefit would be that is unlikely that "one size fits all" when it comes to server implementations, e.g. a structure server using a file-based storage suited for a small work group or

an individual user, would not be applicable in the context of a large organization with more demanding performance and stringent backup requirements.

This still leaves the issue of clear and detailed specification. In the author's opinion XML continue to have much going for it — all the original advantages are still valid. OHSWG are however not the only party that has discovered that DTDs are insufficient for detailed specification. Currently several standards are under development to extend XML with a precise specification language. XML Schema [8] is an ambitious W3C standard currently in Draft state. XML Schema will allow the precise specification of data types and attributes as well as the definition of new data types. As such, it can form an excellent specification language for future versions of OHP.

It would better still, if we for future work could leave the highly detailed world of XML behind, and work with higher level specifications. The Esbjerg CSC tools [18] promises to be such a tool. Using IDL as a specification language, allows developers to work with a more natural level of detail, while developing new services or implementing existing ones. As CSC will have the ability to export DTDs (and conceivably this could be extended to XML Schemas), the low cost entry point of XML is maintained if a developer should desire to use XML on the wire. Furthermore by distinguishing between specification (IDL) and actual protocol, it is possible to support new protocols, such as CORBA, by implementing the changes in the CSC tools.

A modest proposal for future standards submitted to the OHSWG would be to accompany all specifications with open source code, so that other members rapidly could use the new standards, if they should so desire. Indeed, the open source nature of Construct, CSC, and to come, Arakne, are steps in such a direction.

4.2 The existing data model

The OHP data model has been designed to be extensible. Through the use of key/value attributes, it is possible to customize the existing data types to most applications. Again this is something that works well within one implementation of OHP, but makes it difficult to export e.g. the LocSpecs of one OHP implementation to another, as the exact content of the LocSpec is left to the individual implementation. The OHSWG spans a wide field of hypermedia usage and media types and it is probably not realistic to expect a complete standardization on the content of e.g. LocSpecs. However, establishing shared standards for LocSpecs in common media types such as Web pages, XML documents, and bitmap images would facilitate one of the original goals of the OHSWG, namely the sharing of third-party application integrations. In this area we should also not hesitate to adopt standards such as XPointer [5] as we see fit.

4.3 An OHP interchange format

The author suspects that most of the hypermedia systems created by the members of OHSWG have an interchange format in one form or another — it is

certainly the case for Webvise [11], Arakne, and Chimera [1]. There has previously been a call for an OHP interchange format [10], and at Aarhus we have developed and used an interchange format based on OHP (essential following Darmstadt minus the operations). This format has since been extended and is documented as the Open Hypermedia Interchange Format (OHIF) in [11]. A shared OHP interchange format would benefit us with regards to interchange, but it would also make an essential part of a future IETF/W3C OHP standard.

Interchange formats have also other uses. As a part of Coconut project of which the author is a member, a guided tour tool (Ariadne, which also is a part of Arakne) were changed into an applet and deployed on a Danish high-profile Internet portal `http://www.opasia.dk/`. The purpose was to give ordinary Web surfers access to guided tours authored by the editorial staff at the portal. One of the major concerns when designing this applet was scalability and a solution based on a structure server was ruled out partly because of scale, partly because of firewall concerns, and partly because of the Java applet sand box limitations. The generated guided tours were instead exported into a XML interchange format, which then could be easily downloaded from the portal. The system received wide use (at least 15.000 users had by January 2000 downloaded and used the applet). This is an illustration of a case where the interchange format is actually the main target and not some intermediary file.

5 The way to proper standardization

It will avail us naught that we create an excellent open hypermedia standard if no one notices or uses it. At least not if we desire more than publication opportunities. Furthermore a standard based "only" on the collaboration of a bunch of academics is not likely to gather much steam in the IT industry.

As suggested by E. James Whitehead Jr. at the OHSWG 5.5 meeting the time has indeed come for us to disseminate OHP outside OHSWG by creating a "proper" standard.

That there is room for open hypermedia standard is supported by the emergence of XLink [7] — a standard for (largely) navigational hypermedia in (mainly) XML documents. The XLink community has in the context of XML been working with many problems familiar to the OHSWG. At this point the members of the OHSWG have an established infrastructure, we support more than navigational hypermedia, we have integrated numerous third-party applications, we have the know-how to integrate more, and we are not limited to the Web. If we can make our varied and sophisticated hypermedia applications interoperate through OHP, and provide others with the technology to do the same, we will have a strong demonstration of what OHP is capable of.

The first step to a proper standard will however be to establish something that we can present to the world, possibly using the tools described in section 4. Given that, it should be relatively painless to create a RFC. Such a standard should consist of at least an interchange format (see section 4.3), as well a specification of the operations that can performed on the hypermedia structures. In

other words, an extended version of Darmstadt. We are already close to this target, and it would be a pity not to go further. The worst-case scenario would be that our proposed standard is ignored by the rest of the world, but then at least we will ourselves have benefited from the clarification of pinning down the standard.

Beyond RFC, we must consider which standard body our standard belongs to — obvious choices are IETF or W3C, but the discussion of that should be deferred to the time we have created an actual RFC.

6 Conclusion

One of the purposes of this position paper has been to document the feasibility of adopting OHP, and the advantages that it gives. In order to stop OHP from becoming just yet another hypermedia protocol, it is important that the momentum is kept and that other hypermedia systems either convert or interface to the OHP. In that process we must further specify and standardize OHP. This will benefit us double: it will allow systems and users to interoperate and interact (and thus realizing the original goal of sharing third-party application integrations), and it is a showcase for the validity of the work of the Open Hypermedia Working Group. We must practice what we preach, if we are to be taken seriously.

7 Post Scriptum

As a result of OHSWG 6.0, the author has committed to lead in the creation of an OHP RFC. The creation and discussion of this important document can be expected to form a major part of the the OHSWG 6.5 meeting. As of this writing the contents of the RFC can be expected to contain at least the following:

- The navigational data model (preferably also composite and spatial)
- OHP-Nav (ditto OHP-Comp and OHP-Space)
- Session data model
- OHP-Session
- A mapping to protocols (XML, CORBA) based on the Esbjerg CSC work [18]
- An interchange format (presumably an extension and generalization of OHIF [11].

Acknowledgments

The author is a member of the Coconut project (`http://www.cit.dk/coconut/`), a joint research project consisting of Department of Computer Science, Aarhus University and Tele-Danmark Internet. The Danish National Center for IT-Research (`http://www.cit.dk/`) supports the Coconut project. The author wishes to thank René Thomsen, Michael Bang Nielsen, and Henning Jehøj Madsen for their work on Arakne, and Kenneth M. Anderson for valuable discussions.

References

1. K. M. Anderson. Integrating open hypermedia systems with the World Wide Web. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8th ACM Hypertext Conference*, pages 157–166, Southampton, UK, Apr. 1997.
2. N. O. Bouvin. Unifying strategies for Web augmentation. In *Proceedings of the 10th ACM Hypertext Conference*, pages 91–100, Darmstadt, Germany, Feb. 1999.
3. N. O. Bouvin. Designing user interfaces for collaborative Web-based open hypermedia. In *Proceedings of the 11th ACM Hypertext Conference*, pages 230–231, San Antonio, USA, May 2000.
4. N. O. Bouvin and R. Schade. Integrating temporal media and open hypermedia on the World Wide Web. In *Proceedings of the 8th International World Wide Web Conference*, pages 375–387, Toronto, Canada, May 1999. W3C.
5. R. Daniel, S. DeRose, and E. Maler (editors). XML Pointer Language (XPointer). W3C Working Draft 6 December 1999, W3C, Dec. 1999. http://www.w3.org/TR/xptr.
6. H. C. Davis, A. Lewis, and A. Rizk. OHP: A draft proposal for a standard open hypermedia protocol. In *Proceedings of the 2nd Workshop on Open Hypermedia Systems*, number 96-10 in UCI-ICS Technical Report, pages 27–53, University of California, Irvine, USA, 1996.
7. S. DeRose, E. Maler, D. Orchard, and B. Trafford (editors). XML Linking Language (XLink). W3C Working Draft 21 February 2000, W3C, Feb. 2000. http://www.w3.org/TR/xlink/.
8. D. C. Fallside (editor). XML Schema part 0: Primer. W3c working draft, W3C, Feb. 2000. http://www.w3.org/TR/xmlschema-0/.
9. K. Grønbæk, N. O. Bouvin, and L. Sloth. Designing Dexter-based hypermedia services for the World Wide Web. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8th ACM Hypertext Conference*, pages 146–156, Southampton, UK, Apr. 1997.
10. K. Grønbæk and L. Sloth. Supporting interchange of open hypermedia structures and contents. In U. K. Wiil, editor, *Proceedings of the 5th Workshop on Open Hypermedia Systems*, number CS-99-01 in Technical Report, pages 34–37. Aalborg University Esbjerg, Denmark, Feb. 1999.
11. K. Grønbæk, L. Sloth, and N. O. Bouvin. Open hypermedia as user controlled meta data for the Web. In *Proceeding of the 9th World Wide Web Conference*, pages 553–566, Amsterdam, Holland, May 2000. W3C.
12. K. Grønbæk and R. H. Trigg. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. In *Proceedings of the 7th ACM Hypertext Conference*, pages 149–160, Washington DC, USA, 1996.
13. F. G. Halasz and M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, Feb. 1994.
14. J. Jühne, A. T. Jensen, and K. Grønbæk. Ariadne: A Java-based guided tour system for the World Wide Web. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998. W3C.
15. R. Killough and J. J. Leggett. Hypertext interchange with the Dexter model: Intermedia to KMS. TAMU-HRL 90-002, Hypertext Research Lab, Texas A&M University, Aug. 1990.
16. C. C. Marshall and F. M. Shipman III. Spatial hypertext: Designing for change. *Communications of the ACM*, 38(8):88–97, 1995.

17. O. Reinert, D. Bucka-Lassen, C. A. Pedersen, and P. J. Nürnberg. CAOS: A collaborative and open spatial structure service component with incremental spatial parsing. In *Proceedings of the 10th ACM Hypertext Conference*, pages 49–50, Darmstadt, Germany, 1999.
18. U. K. Wiil, P. J. Nürnberg, D. Hicks, and S. Reich. A development environment for building component-based open hypermedia systems. In F. M. Shipman, III, editor, *Proceedings of the 11th ACM Hypertext Conference*, pages 266–267. ACM, May 2000.
19. U. K. Wiil and E. J. Whitehead Jr. Interoperability and open hypermedia systems. In U. K. Wiil, editor, *Proceedings of the 3rd Workshop on Open Hypermedia Systems*, number SR-97-01 in CIT Scientific Reports, pages 137–145, Apr. 1997.

Enabling Project Awareness and Intersubjectivity via Hypermedia-Enabled Event Trails

Kenneth M. Anderson
Department of Computer Science,
University of Colorado, Boulder
ECOT 717, Campus Box 430,
Boulder CO 80309-0430, USA
E-mail: kena@cs.colorado.edu

Niels Olof Bouvin
Department of Computer Science,
Aarhus University
Aabogade 34, DK8200 Århus N,
Denmark
E-mail: n.o.bouvin@daimi.aau.dk

ABSTRACT
Supporting project awareness in the context of large-scale software development is difficult. One key problem is identifying appropriate abstractions and techniques for the insertion of project awareness mechanisms into a software development environment with minimal impact. An additional problem is scaling project awareness mechanisms to handle the demands of large-scale software development projects. We present a framework to support awareness and intersubjectivity among software team members through the use of automatically collected, hypermedia-enabled event trails. Event notification and open hypermedia concepts, techniques, and tools are used to support the framework in addressing the two problems identified above. A distinction of the framework is the presence of mechanisms that explicitly support intersubjectivity among team members, enabling a high degree of quality to the project awareness in the team.

Keywords
intersubjectivity, event trails, project awareness, hypermedia, large-scale software engineering

1 Introduction

Software engineers in modern software development projects are challenged by overwhelming information management tasks. These tasks include, but are not limited to: requirements traceability, consistency management in the face of change, and maintaining project awareness between team members who may be distributed across both time and space. This paper reports on work designed to address the third task—project awareness—especially within the context of large-scale software development. Project awareness pertains keeping individual team members informed of the overall project progress, and aware of teammates' actions. As Dourish and Bellotti state "... awareness is an *understanding of the activities of others*, which provides a *context for your own activity* [13]." Awareness thus touches on issues of sharing information between team members such that they can effectively coordinate the activities of their group.

To gain insight into the scope of information management tasks faced by large software development projects, consider recent work in supporting requirements traceability tasks at a major aerospace corporation [2]. In this instance, open hypermedia techniques [27] were applied to just two subsystems of an avionics software package. These two subsystems consisted of approximately 34,000 pages of documentation used to document various aspects of their hundreds of thousands of lines of code. For the requirements traceability tasks, the aerospace engineers were interested in just six different types of relationships, but on a system of this size, over 500,000 instances of these relationships needed to be created and managed by the open hypermedia system!

More relevant to the domain of this paper, consider the number of personnel that can be assigned to large-scale software development projects. In *The Mythical Man-Month*, Fred Brooks states that, at one point, over one thousand employees were assigned to his project to develop the OS/360 operating system [9]. Modern projects addressing more complex problem domains will, of course, have similar or greater staffing levels but with different characteristics in terms of distribution. All one thousand members of Brooks' team were co-located at the same facility, whereas modern projects are much more likely to be distributed across several physical sites of the same organization or across multiple organizations (as is typical in the aerospace domain).

We therefore are concerned with two issues with respect to project management. First, how can project awareness mechanisms be inserted into the development environment of large software organizations? What infrastructure is required and what techniques (and their associated tools) are enabled by the infrastructure? Second, how can project awareness mechanisms be scaled such that they can provide utility in large-scale software development contexts?

With respect to the first issue, we develop a framework that makes use of techniques from the fields of open hypermedia and event-based messaging systems to achieve *intersubjectivity* between team members. We call this framework *iScent* (intersubjective collaborative event environment). Intersubjectivity can best be defined by the phrase "I know that you

know that I know." It is difficult to communicate without intersubjectivity, since humans typically need to know that they are being understood before proceeding with a conversation. In our framework, we view the actions performed by software engineers as elements of a conversation. We provide concepts and mechanisms to enable intersubjectivity to be achieved in that conversation, thus supporting awareness. With respect to the second issue, we employ a variety of techniques and strategies to construct a scalable implementation of our proposed framework.

The rest of this paper is organized as follows. In the next section, we discuss related work. We then describe our unique approach to the problem of project awareness in Section 3. Section 4 describes our initial attempt to implement our framework. Section 5 describes our future research plans. We then conclude the paper with a summary of our contributions in Section 6.

2 Related Work

In this section, we cover related work from several disciplines including open hypermedia, event notification systems, process-centered environments, awareness support, and document management systems.

Open Hypermedia

In open hypermedia systems, issues of collaboration have been addressed along a variety of dimensions including the collaborative creation of hypermedia structures [18], concurrency control in collaborative hypermedia systems [34], hypermedia services in shared workspaces [35], and the support for collaboration in Web augmentation systems [7, 8]. Typically, support for awareness is derived from event notification facilities contained in collaborative hypermedia systems. These facilities notify all clients of the actions of the hypermedia system's users. Thus, two users in a collaborative session will be notified whenever either user creates a link, deletes an anchor, etc. The Arakne environment, a system for Web augmentation, [7] provides support over and above simple event notification for its users by adding the notion of coupling modes [12] within a collaborative session. While these services in collaborative hypermedia systems help to provide awareness with respect to the hypermedia-related actions of members of a software team, they provide little insight into other aspects of the team's work. In contrast, the iScent framework is designed to support the capture of multiple types of events covering a wide range of work activities. We are currently in the process of integrating our hypermedia systems, Arakne [7] and Chimera [3] into the iScent framework such that we can capture the hypermedia-related activities of software teams within iScent trails.

Event Notification Systems

The iScent framework builds on top of the services provided by event notification systems (See Section 3 for details). In fact, as will be explained in Section 3, the iScent framework is independent of any particular event notification system, such that an implementation can make use of any service that meets the requirements specified in Section 3. Event notification systems were first employed to support tool integration in software development environments. One of the first systems to employ this approach in a local-area network setting was Field [29]. Tool integration via events was also a part of Hewlett Packard's SoftBench environment [10]. In recent years, event notification systems have been extended to explore issues related to events across wide-area networks [11] and enabling project awareness [15, 16, 28].

With respect to the latter, each of these systems adopt a similar approach to iScent in which an event notification service serves as infrastructure to a higher level set of services. For instance, NESSIE [28] and Elvin [15] each use events generated by applications (and sensors in the case of NESSIE) to provide project awareness through mechanisms such as a ticker tape application in which event notifications scroll across the bottom of a user's screen. The iScent framework differs slightly from these approaches in that it provides a unifying abstraction around which project awareness is conveyed (event trails) and it provides additional mechanisms (as explained in Section 3) to enable intersubjectivity. For instance, with a ticker tape application, a user knows that his events are being broadcast to other users in his or her workgroup. However, they do not know if the users have actually seen these events scroll by, perhaps because they were away from their desk at the time or the ticker tape application's display was covered by some other application window. In iScent, several mechanisms combine to achieve the "I know that you know that I know" quality of intersubjectivity: namely that a user's event has been delivered, its recipient has seen it, and the recipient knows that the sender knows that he or she has seen it. These explicit mechanisms insure that intersubjectivity has been achieved and thus raises the quality and fidelity of the project awareness among iScent users.

Process-Centered Software Environments

Process-centered software development environments [1] provide techniques and tools for making the steps of a software life cycle more explicit and visible to the developers participating in a software development project. Recently, we performed an analysis of these environments, using activity theory [25], to examine their support for computer-supported cooperative work [5]. Our analysis, in part, inspected the interaction paradigms employed by process-centered environments. This aspect is relevant to project awareness, since it is through these paradigms that the state of a software life cycle is conveyed to developers. There are three interaction paradigms used in these environments:

- Task-oriented: This interaction style involves the use of *agendas*. Agendas manage lists of relevant tasks for each user, as e.g. in SPADE-1 [4].
- Document-oriented: Interaction is achieved in these

systems via documents and document services. In Merlin [21], for instance, a *work context* graphically displays the relevant documents available to and associated with each user role.

- Goal-oriented: Interaction in this paradigm is centered around a list of goals to be accomplished. In Marvel, for instance, these goals represent currently active rules that can be applied to the state of the process [22].

Each paradigm enables a limited form of project awareness but without the full range of support for intersubjectivity that the iScent framework provides. For instance, few of these systems allow users to specify events such as "notify me when Jane has viewed document A." In addition, the process formalism being applied constrains the type of awareness that can be achieved among team members. At best, they are aware of their process. In contrast, the iScent framework's use of trails of events allow the awareness of multiple aspects of a work environment to be captured, whether or not an event is associated with a particular software life cycle.

Awareness Support

Support for awareness has a rich history of research. For instance, Dourish and Bellotti examined how shared feedback mechanisms can contribute to project awareness among groups organized around a shared workspace [13]. Shared feedback involves the automated collection and distribution of information that is then presented as background information to the participants of a shared workspace. The iScent framework is an example of a shared feedback mechanism in which events are automatically collected and distributed in the background as team members perform work. These events are then presented to users as hypermedia-enabled trails of events. The trails of events can be organized along many intersecting dimensions. For instance, the trail "all events opening document A" can lead to the trail "what did Jane do after she opened document A". Each event in a trail is "hypermedia-enabled," which means that there is enough information associated with an event to allow an open hypermedia system to link the event with its associated application and/or artifacts. In this way, the shared workspace is defined by the event trails themselves; any application or artifact that can be reached from an event is part of the workspace. This allows workspaces to fluidly expand or contract based on the activities of the software team. And, because trails are persistent, it is possible to "travel back in time" and see the structure of a workspace at any point.

One large class of project awareness research involves the use of video to enable project awareness among the members of an organization. Example work in this area includes Portholes [14] and Montage [32]. iScent represents an orthogonal approach to project awareness from these video-based systems; however an interesting intersection between the two domains would be to integrate a video-based awareness system into an event-based framework like iScent such that the video interactions of team members became a part of the project awareness captured by iScent.

A second class of project awareness research involves developing frameworks for adding awareness mechanisms to collaborative applications. Representative samples of this work include [17, 19, 24, 31]. As discussed below in more detail, our approach to supporting awareness does not involve the integration of awareness mechanisms into tools directly. Instead, the iScent framework makes use of event notification systems to transparently capture and distribute event information generated by applications. The iScent infrastructure can then assemble these events into trails. As such, integrating applications into an event notification system remains an issue. However, this task involves significantly less effort than integrating awareness mechanisms directly into an application, since it avoids issues of modifying an application's user-interface. It can even be achieved when an application's source code is not available by the use of wrappers, as long as the application provides some form of external interface.

A third class of project awareness research involves creating high-level frameworks for supporting awareness or, more generally, collaborative functionality in applications. The goal here is to specify a conceptual framework that applications can adhere to, backed by an implementation of the framework that will automatically enable collaboration through the application's use of the framework. Example work in this area includes [6, 20, 26, 30]. Again, iScent places no restrictions on its participating applications other than the requirement that they be integrated with an event notification system. However, conceptually, iScent is able to address issues present in other conceptual frameworks. For instance, Hayashi et al., present a framework designed to support the sharing of knowledge between people, projects, and places [20]. They represent an activity as a chronological thread of snapshots of the information in a workspace. iScent event trails are not restricted to capturing activities along a temporal dimension only. Events are stored persistently and can be assembled into trails along a variety of dimensions. For instance, a user can request to see Jane's trail of events for the work she did last Wednesday, however they can also request to see those same events organized by the projects she was working on, or by the applications that she used that day, or any other axis that can be represented by the value of an event's fields. In addition, the concepts of people, projects, and places can all be inferred from standard event fields such as the user who generated an event (e.g. "John"), their work location at the time (e.g. "John's laptop"), and the project they were working on (e.g. "iScent documentation").

Document Management

LaMarca et al., take an innovative approach for supporting document-centered collaboration in the *Placeless Documents* project [23]. Here, coordination and collaborative functionality is associated with documents rather than applications. This is enabled by associating *active code* with all

operations that can occur on documents, including reading, writing, deleting, etc. When an operation occurs, the active code can maintain coordination and collaborative constraints by posting event notifications, performing additional operations, and even denying the original operation in the first place for example. LaMarca et al., use this framework to describe an application, *Shamus*, that supports software engineers with implementation activities such as checking code in and out of a configuration management system and automatically compiling code and documentation whenever code is changed. They argue that the placeless documents architecture enables a finer granularity of awareness to be achieved than is normal in software development. For instance, Shamus is able to inform developers working on the same piece of code where exactly each developer is working within a file. Previously, developers could only know, at best, that more than one person is working on the same file at once.

In a similar fashion, the iScent framework is an attempt to increase the granularity of awareness that can be achieved via the use of hypermedia-enabled trails of events. As will be discussed in detail in Section 3, an event can be at any level of granularity ranging from key press events to document events to process level events, depending on the amount of detail provided by its source application. While iScent has made a tradeoff because it depends on application integration—a restriction that is not encountered in the Placeless Documents approach—it then has the ability to capture a wide range of events, including those that are not associated with documents. In addition, by making use of open hypermedia, we allow users to quickly traverse from (a visualization of) an event to its related application and/or documents. This latter feature allows users to access documents in a uniform way as part of a collaborative process enabled and managed by the event trails themselves. Thus, the iScent framework can be seen as another point in the design space first mapped by the Placeless Documents project, because it also does not attempt to place coordination and collaboration functionality into applications. It instead associates this functionality with event trails as a new abstraction for project awareness.

3 Approach

We now present the iScent framework. Since the transport layer of iScent is an event notification system, we begin our presentation with a brief review of the capabilities and characteristics of these system. We then present the architecture of the iScent framework and describe in detail its constituent parts. The utility of the iScent framework is illustrated via a scenario which includes a step by step explanation of how iScent enables intersubjectivity. We then conclude this section by briefly addressing privacy concerns raised by the iScent approach.

Event Notification Systems

iScent uses an event notification system as a transport layer. These systems are, in general, based on the concepts of events, producers/consumers of events, and event subscriptions/notifications. Typically, an event is a set of key/value pairs. Producers publish events to an event server which routes these events to consumers based on their subscriptions. One benefit of this arrangement is that producers are completely unaware of the location of interested consumers (and are thus not dependent on these consumers in any way). Likewise consumers are unaware (and not dependent on) producers. This arrangement can lead to significant benefits. For instance, the C2 architectural style makes use of these characteristics to provide substrate independence in software architectures [33]. Other advantages include:

- Producers and consumers focus only on events meaningful to them. They have no need to understand the entire event space being managed by the event system. They are, thus, straightforward to create and configure.
- Event systems make efficient use of a network. When an event is produced, only those consumers who subscribed to the event are notified.
- If several event servers are used (which is, for instance, possible with the Siena system [11]), the routing of events can be further optimized (e.g. an event is only sent to an event server if it has clients that are interested in that event). This facilitates the use of an event notification system across a wide-area network.
- The publish/subscribe model enables dynamic service discovery. For instance, a consumer can publish an event requesting a specific service. If there is a producer that provides the service, it will notify the consumer, and the consumer can subscribe to it.

By making use of an event notification system, the iScent framework provides these same advantages and characteristics to its users.

The Architecture of the iScent Framework

The architecture of the iScent framework is presented in figure 1. The components of the framework are:

iScent Client Applications iScent applications produce iScent events.

Sink A sink consumes all iScent events. It is responsible for making events persistent and provides an interface that allows other components to issue queries over the stored events.

Trail Viewer A trail viewer specializes in the visualization and structuring of collections (“trails”) of iScent events. A trail viewer provides hypermedia capabilities over its trails such that software engineers can traverse from an

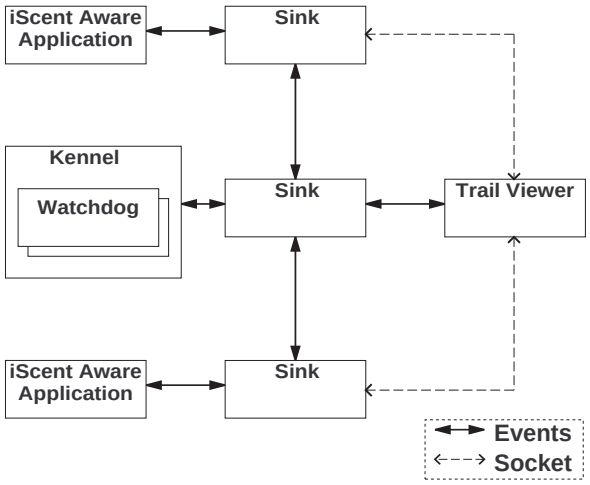


Figure 1: The Architecture of the iScent Framework

event to the event's associated information. These hypermedia capabilities are provided via integration with an open hypermedia system [27].

Watchdog A watchdog is a persistent trigger than can generate iScent events when a specified condition is met. Watchdogs are used to support awareness since they play a key role in achieving intersubjectivity (as described below).

Kennel A kennel is a collection of watchdogs. A kennel will often be associated with a sink, however they can also exist independently.

Communication between iScent components is primarily handled by iScent's integrated event notification system. Communication via events is symbolized in figure 1 by solid arrows, i.e. no point to point network connections exist between components connected by solid arrows. Dashed arrows signify point-to-point socket connections, which are used when large packets of information must be transferred between components. Event systems typically limit the size of events to ensure adequate performance. Therefore, queries with large result sets must be transferred outside the event system.

A typical iScent configuration consists of one or more iScent applications deployed on a set of user machines, and a set of sinks and kennels distributed across a set of server machines. Note, however, that a user can install a sink on his or her local machine. The main benefit of this configuration is the ability to use iScent "unplugged," i.e. without contact with other sinks or event servers. Once a machine is plugged into the network, the local sink can offer its stored events to other sinks for replication and distribution to other team members.

The configuration of iScent components involves linking each component to a local event server and, in the case of sinks, specifying the types of iScent events to store. An event server will likely be coupled with other event servers which are in turn connected to other sinks and iScent applications. It is thus possible to create highly-distributed large-scale networks of iScent components.

An iScent Scenario

The utility of the iScent framework is now demonstrated via a scenario. While the iScent framework is general enough to be applied in many work settings, our first target setting is the task of software development.

> Jane, a system developer, has just returned from vacation. She begins her first day of work by starting a trail viewer. While she has been away, many events matching her criteria have been generated. The trail viewer retrieves these events, and presents them in juxtaposed trails sorted according to topic and time of occurrence. There is too much information to digest, so Jane engages a set of filters that she has previously defined, which simplifies the view considerably by folding matching patterns of events into meta events. Jane sees in the CVS trail that several new files have been checked into a CVS repository. Checking her e-mail, she notices a message from John telling her about one of the new files and asking her to take a look at its associated design rationale in the documentation. She clicks on the event representing the check-in of the file to see further details, and notices a link to the file's documentation. She follows the link which loads the documentation into a Web browser. While reading, she notices that John has put a watchdog on the documentation, and five minutes later, John calls to hear her opinion of the design rationale.

This scenario illustrates key aspects of iScent's functionality. Since iScent events are persistent, it is easy for Jane to come up to speed with what has happened while she was on vacation. Jane need not worry about specifying where events should be retrieved; this is handled automatically by iScent (while she was away, the entire structure of sinks could have been reorganized without her noticing or caring). Rather than presenting a single, large list of events, the presentation of events is structured, and can be further manipulated. To provide further structuring mechanisms, it is possible to create links to and from the events stored in the sinks. Because Jane's Web browser is iScent aware, her viewing of the documentation creates an iScent event, which can trigger a watchdog. This allows John to wait until Jane has actually discovered and viewed his changes, before contacting her.

Enabling Intersubjectivity

We now parse the scenario into single iScent actions to illustrate how the iScent framework enables intersubjectivity among software developers.

When John checked in a new file, an iScent event representing his action was created and stored in a project-related

sink. The creation of the code's documentation also generated iScent events, and according to good practice, John created a link from the check-in event to the documentation using his trail viewer. Because he knew that Jane had expertise on the topic of the new file, he sought her opinion of the design through e-mail. Wanting to be told when she had read the documentation, John used his trail viewer to create a watchdog, set to trigger when Jane retrieved the documentation with her Web browser. This watchdog was stored at a project-related kennel, and the kennel issued a subscription matching the criteria of the watchdog. By posting the watchdog, John automatically created a subscription matching the event that the watchdog would send if triggered. To use the iScent vocabulary, John wanted to be alerted if his and Jane's trails of actions intersected at the point of the documentation. Having done this, John was free to work on other things, knowing that he would be notified when to contact Jane.

When Jane started her trail viewer, the trail viewer issued a series of query events on the topics she had configured it to follow (essentially the subscriptions she had created earlier). In this case, Jane was interested in iScent events of the types "CVS" and "Documentation." These query events were consumed by a project-related sink which then confirmed that it could answer her query. The confirmation events held information (a network address and a port number) for the trail viewer to retrieve the trails generated by the sink based on the queries. The trail viewer proceeded to contact the sink and retrieve the trails.

Once retrieved, the trail viewer displayed the trails, and Jane could begin to manipulate them to gain an overview of the changes made while she was on vacation. In this situation, the trails were arranged by type ("CVS" and "Documentation") along a time axis. When Jane read the e-mail from John, she found the matching event, and followed a link from it to the documentation. Following the link caused the documentation to be loaded into Jane's Web browser. Since the Web browser was iScent-aware, it generated an iScent event that recorded the action of loading the documentation. At the project kennel, John's watchdog was triggered by this event. The watchdog, in turn, generated an event containing its ID and the information about the person, who had triggered it. Because a trail viewer by default subscribes to events that contains a "triggered by" field matching its user, the event was received by Jane's trail viewer, alerting her to the existence of a watchdog and to the identity of its creator. Simultaneously, the same event was received by John, and he now knew that he could contact Jane, in the near future, to discuss the design rationale.

By reading the email, Jane knew that John wanted her to read the documentation. When she actually read the documentation, John was alerted, so that "he now knew that she knew". Conversely, Jane knew by being alerted by a watchdog, that "John knew that she now knew" about the design rationale. Thus, Jane was aware of the context for John's call because intersubjectivity had been achieved.

Trails

Trails are a cornerstone of the iScent framework. Through the use of iScent applications, a user generates a trail of iScent events (a "scent trail"). These are stored by sinks, and can later be retrieved through queries. A trail is defined as a set of iScent events. Thus, the actions of a user can be captured by a trail of iScent events matching the user over time. Another trail could be the events matching a certain tool over time, or following the history of a document. These trails intersect, for instance, when a user makes use of a certain tool, or modifies a certain document. Since trails are modeled as sets of events, trails support the normal set operations: join, intersection, and difference. These operations match on selected common fields among the constituent events, and can be performed by a trail viewer or by expressing a query to reflect the desired operations.

The user manipulates trails in a trail viewer. Trails can be plotted along axes defined by the user. Returning to the scenario, Jane plotted two trails consisting of CVS events and documentation events respectively along a time axis in order to see the correlation between code produced and documentation written. When plotting a trail, one field common to all the constituent events is used to order the events along an axis. As a time stamp is common among all iScent events (see Table 1 for the default iScent fields), a common ordering is time based, though other orderings can be used.

As a set of events, a trail can be arbitrarily large. One of the design goals of iScent is to produce highly detailed events. However in some circumstances too much detail can be a liability, since a user can easily lose his or her sense of the "big picture" behind the events. To address this, without abandoning high fidelity events, trails can be filtered. A filter is essentially pattern matching on events, such that matching events are replaced in a trail viewer with a meta event (or a "folded" event). One could for instance define a pattern to replace a sequence of CVS events relating to a single file with a single event representing all actions pertaining to the file. The user can of course unfold a folded event to inspect the constituent events. Once defined, filters are stored in a trail viewer and can be applied as the user desires. Folded events are iScent events and can, as such, also be subjected to filtering.

Trails of events can be transient, e.g. existing only on a user's screen, or they can be stored — either as a specification of the queries and filters that produce the trail, or as a sequence of event ids (all iScent events have unique ids, as described in Section 4). Finally, it is possible to export (or import) trails as XML files for external use. The export format is identical to the format used by sinks to send trails to trail viewers.

Privacy Concerns

Not all actions taken by an iScent user is necessarily relevant for other users. Sometimes it makes sense to track all ac-

tions, such as a user's CVS operations, and sometimes it does not, such as what Web pages a user visited. User can and should therefore be able to configure filters that designate the types of events they want to publish. In the context of a Web browser, one could, for instance, publish all events regarding technical documentation. This not only addresses privacy concerns, but also improves the overall signal-to-noise ratio in the trails stored by sinks.

4 Implementation

We have constructed a prototype implementation of the iScent framework, including initial implementations of the following components: sinks, watchdogs, and kennels. These components are sufficient to test the creation and storage of iScent events and to validate the iScent type system and query facility. Our work on a trail viewer is preliminary and has involved the design of its user interface and the construction of a tool that can query and retrieve event trails from sinks. Our initial prototypes have been stress tested through the use of batch tools that generate tens of thousands of events and then issue queries that validate the operation of the sinks, watchdogs, and kennels. These initial efforts represent a proof-of-concept of the iScent framework; our future work will involve fleshing out the trail viewer and integrating client applications. Below, we discuss the various issues that our initial implementation has raised.

Requirements of the Event Transport Layer

The iScent architecture relies on an event notification system to provide the transport layer for iScent events. The requirements on the transport layer are:

- Events must consist of key/value pairs
- Values can be arbitrary strings
- Subscriptions to events must be supported
- Subscriptions to events should be specific key/value pairs (e.g. "name equals John Smith" or "age greater than 24")

These requirements are met by most event notification systems. Therefore, the iScent framework is independent of any particular event system. The current version of iScent utilizes the Siena event notification system [11], which easily satisfies the above requirements.

Scalability of iScent

iScent's scalability is dependent on the scalability of its associated event notification system. Our implementation of iScent therefore depends on the scalability of Siena, which has been characterized by Carzaniga *et al.* in [11]. One of the main benefits of Siena is that Siena servers can be coupled together in a network. Given a situation where many iScent components and Siena servers are in use, Siena automatically provides optimal routing of events from producer to consumer to maximize performance and minimize latency.

Multiple Sinks

iScent can be configured with any number of sinks. This has several advantages over a single store:

- Sinks can be local to a workgroup, and run on a local machine. This provides improved performance as the sink only stores events produced by its workgroup.
- While maintaining local sinks, the addition of larger sinks collecting all events generated by, e.g. a department, provides additional benefits. Data is automatically replicated between sinks for greater data security, and if one sink is down or slow due to high load, other sinks can provide access to the desired events.
- These configurations are achieved by installing more or less specialized subscriptions into sinks that determine the set of events they store and share. A configuration is flexibly manipulated since a sink can be added or removed *at run-time* simply by changing its subscriptions.

Querying for Events

Sinks provide for the persistence of events and handle all queries. Queries are sent to sinks via Siena. This allows queries to be handled in an efficient and scalable manner.

A query is an iScent event that specifies the type of desired events and (ranges of) values in these events. A sink subscribes to the query events matching what it stores, and queries are thus automatically routed (by the underlying transport layer) to the sinks that can provide answers. When an iScent component issues a query event, it also subscribes to confirmation events matching the query. If there are sinks that are able to fulfill the query, they will issue confirmation events which are then received by the querying party. While most event notification systems can efficiently route events, they are in general not optimized for very large events (Siena for instance has an upper limit of 64 K). Query results can be arbitrarily large, and rather than returning the query result itself in an event, the confirmation event contains contact information for the answering sink. Upon retrieval of the confirmation events, the querying party establishes direct socket connections to the sinks and retrieves the events (it can also choose to select only one of the sinks for retrieval). This lessens the load on the event transport layer considerably. To lessen the network load further, the queries are compressed and decompressed automatically. The retrieval of query results is the only time where an iScent component makes a direct point to point connection to a sink — at all other times it generates iScent events that are automatically propagated by the event transport layer.

The query and confirmation events are themselves iScent events, and are, as such, stored by the sinks. Apart from the awareness possibilities in these events, it also allows a system administrator to monitor how the sinks are used, and

exploit this information to move sink data and subscriptions to improve network performance.

A sink uses an SQL database for storage, and automatically generates a new table, when an iScent event of a new type (see below for information on types) is received. This facilitates rapid query resolution. The current query interface has been designed to make the most of the SQL database. Queries can be combined using logical operators, and is able to create very precise queries.

iScent Events
An iScent event is the atom of the iScent framework. An event is composed of key/value pairs (or "fields"). All events have a default set of key/value pairs, shown in Table 1.

Key	Example Value
User	John Smith <john@comp.com>
Producer	org.iscent.app.iScentApp
Host	138.128.34.10
Timestamp	956809312389
id	9KjxG3iBMCvHALTrOe=6xW
Types	type1 type4

Table 1: The default iScent key/value pairs

Thus, an event can provide information about its user, how it was created, where it was created, and when. The id is a 128 bit integer (here presented in `base64` form), created as a MD5 hash signature of the rest of the event. The use of MD5 ensures both a unique id and data integrity (since the MD5 signature is recomputed at arrival and compared with the existing one). In addition, most events use the Types field to declare their type(s), i.e. specifying which other fields the event contains.

The iScent Type System
iScent events are typed, and are checked upon creation and arrival. The type system is component based and extensible. Types are defined through iScent type declaration events and are stored by sinks. A type declaration consists of a (globally unique) type name, one or more fields with key names unique within the type, and a definition of the types of these fields.

All values in an event (with the exception of the default values listed in Table 1) are encoded in XML. The field definition is a DTD specifying the format of legal values for that field. The DTDs are stored by sinks and are used by them to validate the events they consume.

The basis of the iScent type system is the declaration of single types, but iScent events can have more than one type. When an event declares several types, it guarantees that it contains valid values for all the fields defined in all of the types. The names of keys are a combination of their original iScent type name and their own field name, which eliminates name collisions between types. Combining types can be very useful; a class of iScent applications can, for instance, share a common "header" type, plus their own specialized types. As the number of iScent applications grows, so do the types defined. These types then provide building blocks that can be used by other iScent developers when constructing new iScent applications.

Watchdogs and Kennels
A crucial part of supporting intersubjectivity is the watchdog. The watchdog is used to detect the occurrence of certain iScent events (for instance, opening a document), and alerts the creator of the watchdog when the condition has been met.

Currently a watchdog consists of two parts: criteria that must be met, and an event to send when triggered. Watchdogs can be set to trigger only once, and then disappear, or to continue to trigger when conditions are met. If a watchdog reported only to its creator, it would increase the creator's project awareness but it would not provide intersubjectivity, since the people triggering the watchdog would be unaware of its existence. To enable intersubjectivity the watchdog also reports to the person who has triggered it. Trail viewers automatically subscribe to watchdogs events triggered by their user, and will alert its user when such an event is received. Likewise, by creating a watchdog, a user automatically subscribes to responses from it. These responses will be received and displayed by a user's trail viewer.

To ensure the persistence and vigilance of watchdogs, they reside in kennels. A kennel is a server that registers the conditions of its watchdogs and creates matching subscriptions. When an iScent event is received, it is checked against the conditions and, if a match is found, triggers the creation of an event specified by the appropriate watchdog.

The triggering mechanism is currently a simple if-then mechanism. Future work will involve the creation of more sophisticated watchdogs. The most likely approach for improvement will be the creation of watchdog Java classes, that can be uploaded into a kennel. These Java watchdogs can maintain state between triggering events, so they can, for instance, be used to detect patterns of events and generate an event only after a pattern has been detected. Another topic for future work is the migration of watchdogs. Currently watchdogs are sent to the nearest kennel (or rather the kennel that first accepts the watchdog event). To optimize performance and minimize the network load, it would be better if watchdogs could migrate to a kennel close to the producers of events that match the watchdog's criteria.

5 Future Work
There are many avenues for future work in further developing the iScent framework and its associated implementation. Chief among them is the need for evaluating the iScent framework's ability to address the project awareness needs of modern software development projects. We plan to perform this type of evaluation with two very different methods. The first method is to conduct laboratory-based usability studies

on the user interface of the trail viewer and the kennel. We need to determine if the conceptual model presented by the iScent framework provides utility to software engineers and if the user interface is effective in delivering the functionality of the iScent framework into the hands of its end users. The second method is to conduct field studies of the iScent framework in use at an industrial site. The key problem here is identifying industrial partners and obtaining a commitment to participate in industrial collaboration. We have successful track records in university/industry interchange [2] and are beginning the process of obtaining industrial collaborators to support this line of research. The goal of the field studies will be to increase our understanding of the work practices currently used in industry and how the iScent framework either hinders or enhances these procedures.

Secondary plans for future work on the iScent framework involve exploring techniques to increase the fidelity of event trails from integrated iScent applications. Application events of high fidelity leads to better "conversations" between engineers and increases the intersubjectivity that can be achieved by the iScent framework. However, it must not be difficult to integrate applications into the iScent framework and thus techniques are needed to reduce the effort required to integrate an application into the framework. Again, such efforts will enhance the project awareness that can be achieved by the iScent framework by giving it more sources of events and thus increasing the percentage of project work it captures. In addition, we will be exploring new trail visualizations, new query mechanisms over trails, more flexible support for watchdog behaviors, and better tool support for managing iScent networks.

6 Conclusions

We have presented a framework designed to support project awareness in large-scale software development contexts that takes a unique approach to the problem. Leveraging already existing open hypermedia and event messaging infrastructures, the iScent framework combines the use of hypermedia trails with event publish/subscribe techniques to enable intersubjectivity between software engineers and thus promote wide-spread project awareness throughout a software development team. We have described the conceptual layout of the framework and provided insight into an experimental implementation of it. The implementation employed aggressive reuse of fielded infrastructure, database technology, and distributed system techniques in order to scale the implementation to a level where it becomes feasible to conduct field studies of the new technology in industrial settings. Having met these initial scalability concerns, our attention now turns to evaluating the utility of our formalisms and the effectiveness of our prototypes.

REFERENCES

[1] V. Ambriola, R. Conradi, and A. Fuggetta. Assessing process-centered software engineering environments. *ACM Transactions on Software Engineering*, 6(3):283–328, 1997.

[2] K. M. Anderson. Supporting industrial hyperwebs: Lessons in scalability. In *Proc. of the 21st International Conference on Software Engineering*, pages 573–582, Los Angeles, CA, USA, May 1999.

[3] K. M. Anderson, R. N. Taylor, and E. J. Whitehead Jr. Chimera: Hypertext for heterogeneous software environments. In *Proc. of the European Conference on Hypermedia Technology (ECHT 1994)*, pages 97–107, Edinburgh, Scotland, 1994.

[4] S. Bandinelli, E. Di Nitto, and A. Fuggetta. Supporting cooperation in the SPADE-1 environment. *IEEE Transactions on Software Engineering*, 22(12), 1996.

[5] P. Barthelmess and K. M. Anderson. A view of software development environments based on activity theory. *Computer Supported Cooperative Work*, To appear in the Special Issue on Activity Theory, 2000.

[6] M. Beaudouin-Lafon and A. Karsenty. Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window systems. In *Proc. of the ACM Symposium on User Interface Software and Technology*, pages 171–180, Nov. 1992.

[7] N. O. Bouvin. Unifying strategies for Web augmentation. In *Proc. of the 10th ACM Hypertext Conference*, pages 91–100, Darmstadt, Germany, Feb. 1999.

[8] N. O. Bouvin. Designing user interfaces for collaborative Web-based open hypermedia. In *Proc. of the 11th ACM Hypertext Conference*, San Antonio, USA, May 2000.

[9] F. P. Brooks Jr. *The Mythical Man-Month*. Addison-Wesley, 20th anniversary edition, 1995.

[10] M. R. Cagan. The HP SoftBench environment: An architecture for a new generation of software tools. *Hewlett-Packard Journal*, 41(3):36–47, June 1990.

[11] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an Internet-scale event notification service. In *Proc. of the 19th ACM Symposium on Principles of Distributed Computing*, July 2000.

[12] P. Dewan and R. Choudhary. Coupling the user-interfaces of a multiuser program. *ACM Transactions on Computer-Human Interaction*, 2(1):1–39, 1995.

[13] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proc. of the ACM 1992 Conference on Computer-Supported Cooperative Work*, pages 107–114, Toronto, Canada, Oct. 1992.

[14] P. Dourish and S. Bly. Portholes: Supporting awareness in a distributed work group. In *Proc. of the ACM Conference on Human Factors in Computing Systems*, pages 541–547, May 1992.

[15] G. Fitzpatrick, T. Mansfield, S. Kaplan, D. Arnold, T. Phelps, and B. Segall. Augmenting the workaday world with Elvin. In *Proc. of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 431–450, Copenhagen, Denmark, 1999.

[16] L. Fuchs. AREA: A cross-application notification service for groupware. In *Proc. of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 61–80, Sept. 1999.

[17] L. Fuchs, U. Pankoke-Babatz, and W. Prinz. Supporting cooperative awareness with local event mechanisms: The GroupDesk system. In *Proc. of the 4^{th} European Conference on Computer Supported Cooperative Work*, pages 247–262, 1995.

[18] K. Grønbæk, J. Hem, O. L. Madsen, and L. Sloth. Designing Dexter-based cooperative hypermedia systems. In *Proc. of the 5^{th} ACM conference on Hypertext*, Nov. 1993.

[19] C. Gutwin and S. Greenberg. Design for individuals, design for groups: Tradeoffs between power and workspace awareness. In *Proc. of the ACM Conference on Computer Supported Cooperative Work*, pages 207–216, Nov. 1998.

[20] K. Hayashi, T. Hazama, T. Nomura, T. Yamada, and S. Gudmundson. Activity awareness: Framework for sharing knowledge of people, projects, and places. In *Proc. of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 99–118, Sept. 1999.

[21] G. Junkerman, B. Peuschel, W. Schäfer, and S. Wolf. MERLIN: Supporting cooperation in software development through a knowledge-based environment. In A. Finkelstein, J. Kramer, and B. Nuseibeh, editors, *Software Process Modelling and Technology*, pages 103–130. Research Studies Press Ldt, 1994.

[22] G. E. Kaiser, N. S. Barghouti, and M. H. Sokolsky. Preliminary experience with process modeling in the Marvel software development environment. In *Proc. of the 23rd Annual Hawaii International Conference on System Sciences*, volume Vol. II — Software Track, pages 131–140, 1990.

[23] A. LaMarca, K. Edwards, P. Dourish, J. Lamping, I. Smith, and J. Thornton. Taking the work out of workflow: Mechanisms for document-centered collaboration. In *Proc. of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 1–20, Sept. 1999.

[24] J. C. Lauwers and K. A. Lantz. Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window systems. In *Proc. of the ACM Conference on Human Factors in Computing Systems*, pages 303–312, Apr. 1990.

[25] A. Leontjev. *Activity, Consciousness, and Personality*. Prentice Hall, 1978.

[26] D. Li and R. R. Muntz. COCA: Collaborative objects coordination architecture. In *Proc. of the ACM Conference on Computer Supported Cooperative Work*, pages 179–188, Nov. 1998.

[27] K. Østerbye and U. K. Wiil. The Flag taxonomy of open hypermedia systems. In *Proc. of the 7^{th} ACM Hypertext Conference*, pages 129–139, Washington DC, USA, 1996.

[28] W. Prinz. NESSIE: An awareness environment for cooperative settings. In *Proc. of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 391–410, Sept. 1999.

[29] S. P. Reiss. Connecting tools using message passing in the field environment. *IEEE Software*, 7(4):57–66, 1990.

[30] M. Roseman and S. Greenberg. Building real-time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3(1):66–106, 1996.

[31] M. Sohlenkamp. Integrating communication, cooperation and awareness: The DIVA virtual office environment. In *Proc. of the ACM Conference on Computer Supported Cooperative Work*, pages 331–344, Oct. 1994.

[32] J. C. Tang, E. A. Isaacs, and M. Rua. Supporting distributed groups with a montage of lightweight interactions. In *Proc. of the ACM Conference on Computer Supported Cooperative Work*, pages 23–34, Oct. 1994.

[33] R. N. Taylor, N. Medvidovic, K. M. Anderson, E. J. Whitehead Jr., J. E. Robbins, K. A. Nies, P. Oreizy, and D. L. Dubrow. A component- and message-based architectural style for GUI software. *IEEE Transactions on Software Engineering*, 22(6):390–406, June 1996.

[34] U. K. Wiil and J. J. Leggett. Concurrency control in collaborative hypertext systems. In *Proc. of the 5^{th} ACM Conference on Hypertext*, pages 14–24, Nov. 1993.

[35] U. K. Wiil and J. J. Leggett. HyperDisco: Collaborative authoring and Internet distribution. In *Proc. of the 8^{th} ACM Conference on Hypertext*, pages 13–23, Apr. 1997.

Collaborative Web-based Open Hypermedia and Mutual Awareness

Niels Olof Bouvin
Department of Computer Science, Aarhus University
Aabogade 34A, DK8200 Aarhus N, Denmark
E-mail: n.o.bouvin@daimi.aau.dk

Abstract
This paper addresses some issues in collaborative hypermedia and how hypermedia can be augmented with recent advances in awareness technologies, especially event notification systems. The work done in the Open Hypermedia Systems Working Group with special focus on collaboration is presented, along with the Arakne Environment, a hypermedia system based on the Open Hypermedia Protocol. These systems are discussed in the context of the collaboration issues raised.

Keywords
Collaborative hypermedia, open hypermedia, event notification, peripheral awareness, World Wide Web.

1 Introduction

A basic form of collaboration in large document spaces is collaborative structuring and annotation. By structuring a document space, we begin to make sense out of it. From its beginning with Bush's Memex [10], the hypermedia research community has worked with the structuring of document spaces. The Memex was intended to make it possible for knowledge workers to handle the in 1945 rapidly growing scientific literature efficiently by supporting associative thinking. Users of the system would create trails connecting related documents, and these trails could later be shared with other Memex users.

Today, we face the largest document space ever created: the World Wide Web [6]. The Web owes much of its success to its simple architecture, yet this architecture has some limitations, especially with regards to structuring and collaboration. Users cannot create links from Web pages they do not own, nor can they create annotations or other hypermedia structures. Whereas the users of the (admittedly hypothetical) Memex could share trails, the modern Web user is limited to the exchange of URLs. To address this, open hypermedia systems have been extended to provide users with sophisticated hypermedia functionality on the Web. Thus, the basic collaborative act of structuring can be supported.

Another aspect of collaboration is keeping track of coworkers' actions inasmuch as they relate to one's own activities. Collaboration takes place within a shared understanding of the task at hand, and this shared understanding or awareness may be supported by computers. Shared awareness tools range from video systems such as Portholes [15] to messaging systems such as Elvin [18].

On one side, we have hypermedia tools suited for structuring large document spaces, and on the other we have awareness tools suited for helping people collaborate. In the middle, we have the Web.

Based on this context of collaborative hypermedia and shared awareness tools, this paper presents a hypermedia system to augment the Web, both structural and collaborative.

The paper is structured as follows: Section 2 introduces the main inspirations and foundations for this paper. Section 3 discusses collaboration in hypermedia and how this collaboration may be achieved. Section 4 introduces the Open Hypermedia Systems Working Group (OHSWG), and describes the work done by the group with special focus on collaboration. Section 5 introduces the Arakne Environment, a hypermedia system based on OHSWG technologies. Section 6 discusses collaboration in the Arakne Environment in the context of the issues raised in Section 3. Section 7 describes directions for future work, and the paper concludes in Section 8.

2 Related Work

CSCW and hypermedia share common roots. Some of the earliest papers on both hypermedia and collaboration support were Bush's Memex [10, 32] and Engelbart's Augment [16, 17]. These tools sought to improve work by helping knowledge workers to better structure their data and thus better understand their problem area. Memex would allow readers to share trails tying documents together, and NLS/Augment was a distributed, collaborative hypermedia and authoring system, even featuring video conferencing.

Being able to annotate and structure information can be seen as a collaborative activity. While personal annotations certainly are beneficial to the original annotater, they are also valuable to others, even when these annotations were not originally written with the express purpose of publication, as described by Marshall in [33]. By hypermedia structuring, we impose order, we assert relationships where none were before, we admit a text into a larger context. These are

basic characteristics of hypermedia, and characteristics that can only be fully realised in a system where users are free to link, annotate, and structure as they see fit.

The Web is no such hypermedia system. Barring self-publishing, the Web is a read-only medium. Users cannot link from pages, they do not own, and there are no provisions for annotations. The Web is not unique in this aspect. The field of open hypermedia has since 1989 [35, 37] worked on providing users with advanced hypermedia functionality by integrating third-party applications. By hypermedia enabling the tools commonly used, the users gain the advantages of hypermedia structuring without having to abandon their tools of choice. Notable open hypermedia systems include Microcosm [29], HyperDisco [43], DHM [24], Chimera [3], and HOSS [36]. These system extend existing applications so that users can create links and other hypermedia structures between e.g. word-processing documents and spreadsheets. Given this extension of existing systems, it should come as no surprise, that the open hypermedia community has also developed systems to extend the hypermedia functionality of the Web. Systems, such as DLS [12, 11], DHM/WWW [21], Webvise [23], and the Arakne Environment [7], all provide users with the ability to create hypermedia structures on top of arbitrary Web pages. This is accomplished without modifying the original Web pages, and does thus not rely on having write access.

As mentioned in the beginning of this section, some of the earliest hypermedia systems were collaborative in nature. Later systems, such as KMS [1], Intermedia [34], DHM [22], HyperDisco [44], and SEPIA [27, 28, 40] have provided their users with collaboration support for hypermedia authoring.

As described by Heath and Luff [30] in their analysis of work done by traffic controllers in the London Underground, maintaining shared awareness between co-workers can be crucial. Many systems has endeavoured to support this. The GroupDesk system [20], the NESSIE system [38], the Elvin system [18], the AREA system [19], and the BABBLE system [9] to mention a few. These systems try to support shared awareness through means such as messaging, chat, and event notification. On the Internet, commercial "buddy list" programs, such as ICQ or AIM, provide their users with similar functionality.

The Web is increasingly becoming the infrastructure of computing, and as such it becomes natural to collaborate over the Web. An example of a system supporting this is BSCW [5]. To enable collaborative authoring on the Web, the HTTP protocol has been extended by the WebDAV (Web Distributed Authoring and Versioning) working group [41]. WebDAV compliant Web servers provide users with document locking and versioning. With advances such as these, the Web is slowly becoming suited for collaboration.

3 Collaboration in Hypermedia

Wiil and Leggett [42] discuss the requirements of a collaborative hypermedia system. Their six requirements are:

1. Event notification
2. Fine-grained notification
3. User-controlled locking
4. Shared locking
5. Fine-grained locking
6. Persistent collaboration information

This section discusses these requirements as well as other techniques that might advance collaboration in hypermedia systems.

Event notification

Shared awareness is a crucial part of collaborative working. Indeed, shared awareness and mutual understanding forms the foundation without which communication and thus working is rendered highly problematic. They form the basis of articulation work [4], i.e. the process of co-ordinating work. One technique to support shared or peripheral awareness is event notification services. Users generate, either implicitly (e.g. by using an application integrated with the notification system that generate events automatically) or explicitly (e.g. by messaging), events through the use of computer artifacts. Other users can subscribe to these events and are thus kept aware.

Event notification is only as valuable as the information contained within the notification events. Furthermore, the value of an event notification system does not necessarily increase with the amount of information, i.e. received notifications. Many of the actions taken by co-workers are not directly relevant to the task at hand, but some may be crucial. Likewise, there may be only a few notifications that are really important to a user, and if many events are generated and received, the important ones are likely to be overlooked, disappearing in the "noise" of the other events. This problem is usually addressed through a subscription model, where users before hand signify which kind of notifications they are interested in. Depending on the sophistication of the system, a subscription can be simply to an entire class of events, or may specify conditions only a few events will match. Subscriptions cannot solve the whole problem, as a user may create a too restrictive subscription and loosing important events, or create a too inclusive subscription resulting in the signal/noise problem described above. Many of these concerns can be addressed by employing "human filtering" through messaging functionality, so co-workers can alert each other directly. In practice, this is often accomplished through ordinary communication or email.

An interface for notifications championed by the Elvin system [18] is the ticker tape. The ticker tape is a non-intrusive

interface, where notifications scroll by as they are received. The ticker tape is updated over time, as new notifications replace old ones. A user can focus on the ticker tape, when the user wishes without being interrupted in the continuing work and without having to open new windows or use a special interface. A weakness of the Elvin model (also found in other notification services) is that once a notification has scrolled by or has timed out, it is lost, as notifications generated are not stored (they are not "mailbox" notifications in the terms of Wiil and Legget). It is thus most suited to support awareness as work progresses, and cannot be used to catch up with the activities of others.

Collaborative authoring systems as well as collaborative hypermedia systems often uses "live updates", e.g. as one worker makes updates, these updates are propagated to other users working on the same artifacts. This keeps the focus of the main tool used, rather than having the user consult with other tools to see what others are doing. An example of a system using this approach is SEPIA [27, 28, 40], where changes to documents can be broadcast, as they are made.

The live update approach works well with notification services, as users can see that changes have taken place, and see in the notification interface who caused the change, or see comments regarding the changes. A related technology is the use of chat systems or discussion boards used in conjunction with the ongoing work (exemplified by the Babble system[9]). This allows users to engage in conversations and co-ordination over time and location, and, depending on the system, users can refer back to discussions past.

Locking

One of the basic co-ordination technologies found in most collaborative authoring systems is the possibility of locking resources, so that only one person may modify it at a time. A prime example of a general document locking technology that becoming increasingly more widespread is the WebDAV system, described by Whitehead and Goland [41]. By relying on (WebDAV complicant) Web servers for storage, users can issue time-based locks on Web resources, which allows geographical dispersed users to work together on a set of documents without accidently overwriting each others' changes. Locking requires that it is possible to identify the resource to be locked. In the case of WebDAV the atom resource is a single document. This is fairly coarse-grained compared to other systems, where locking can take place on e.g. one paragraph of text. The advantage of the WebDAV system is that it does not require the system to know the semantics of resources stored within it and it can thus be much simpler and is not vulnerable to changes in the internal format of the stored resources. Systems providing finer-grained locking will in general have to use a special application for editing which probably has contributed to the lack of wide-spread use of fine-grained locking, as users are understandably hesitant to abandoning e.g. their word processor of choice "just" to gain fine-grained locking.

In the context of hypermedia systems, locking is also concerned with locking of hypermedia structures such as links and anchors. The need for locking of such resources are however less pressing than the need to be able lock entire documents. One of the classic collaborative hypermedia systems, KMS [1] used optimistic concurrency control of frames. There were no locking mechanisms for frames as it was unlikely that two persons would be editing the same frame simultanously given the large number of frames. The point can also argued from the point of long versus short term exclusive use of a resource. Documents are often large, so users can be expected to spend a relatively long time making modifications, and in that case it makes very good sense to lock document to avoid overwrites. However, when the resource in question is small and it takes a short time to modify it, the need for locking decreases, especially if updates to a resources are propagated to the relevant users, so no one is unaware of the changes made. This is the case of most hypermedia structuring mechanisms. Links and anchors are small entities and the changes to them can be readily distributed.

Persistent collaboration information

Collaboration may but is unlikely to be over in one work setting. Works continues the next day, as does the need for co-ordination and awareness between workers. Collaborative systems should and often do reflect this, so that users need not start from scratch each morning.

Persistent information about collaboration can include information such opened documents, tools used, subscriptions to notifications, and so on. Thus, work can continue from the point where it was left earlier. Another benefit of persistent collaboration information is that it allows the user be engaged in several collaborations at time (or to alternate different collaborations), each collaboration with its own "space" of documents, locks, tools, events, etc.

Modes of Collaboration

Depending on the work situation, it can make sense to differ between modes of collaboration. One situation might dictate that the worked upon resources should be updated across the participants' computers as soon as the changes are made. Some situations may go further with the need for e.g. shared cursors, so that all using the system see the same. On the other hand, a user might want to be temporarily isolated from the actions and updates of others, allowing the user to concentrate at the task at hand without interruptions or distractions. This can be supported in the collaborative application by employing set "policy" as when and how to distribute changes. In the context of notification services, it might be possible to grade events, so that all or only the most important notifications are displaying (thus returning to subscriptions, but in this case with a defined set of subscriptions). It is also possible that the user might want to be engaged in one live session with immediate updates, and still continue working in other contexts with less frequent updates. Thus, working with one task, but being updated from time to time

on the progress on other tasks.

The SEPIA system handles modes of collaboration through different "coupling modes" ranging from uncoupled (a single author working on a document) over loosely (authors collaborating) and tightly coupled (authors seeing each others' updates as they are made) modes. Coupling modes for documents are changed automatically as more or less authors work on a document.

4 The Open Hypermedia Systems Working Group

This section briefly introduces the Open Hypermedia Systems Working Group (OHSWG)[1], and the current state of its work with regards to collaborative hypermedia.

OHSWG is a working group, now in its sixth year, consisting of open hypermedia researchers. One of the goals of the group has been to develop a shared understanding of open hypermedia and to support interoperability between open hypermedia systems. The current result of this effort is the Open Hypermedia Protocol (OHP) [13]. OHP specifies the interaction between hypermedia clients and servers, and has a general and extensible hypermedia data model. OHP has been adopted by the Construct hypermedia servers [45], developed at Aarhus University and Aalborg University Esbjerg, Denmark. In addition to navigational hypermedia, Construct also support compositional and spatial hypermedia

OHP supports collaboration through the notion of sessions. A group of people working together on hypermedia structures on a set of documents using various hypermedia tools embodies the session. A session is defined by a set of users, documents, and tools, and has a coupling mode (ranging from uncoupled to tightly coupled) and a joining police (e.g. a public or private session) associated with it. Through the session, each collaborator is visible to the other collaborators, as are his or her actions, depending on the coupling mode. The visibility of others' actions depends on the coupling mode, as does the degree of latency in updates of the local runtime hypermedia structures. Members of a session are engaged in the same coupling mode and are thus equally visible to each other. Typical coupling modes include:

Uncoupled mode Updates on the structure server are not broadcast to clients. Users save changes explicitly.

Loosely coupled mode Changes on the structure server are broadcast to clients. Changes are stored automatically.

Tightly coupled mode Changed on the structure server are broadcast to clients. Changes on the controlling client are broadcast to other clients, keeping the hypermedia views involved in sync.

If a collaborator so desires, special subscriptions can be set, so that specific actions taken by certain users result in notifications. The session is persistent and users can engage and disengage at any point, so that a group at a later point can resume work where it was left.

To support collaboration a session keeps a session record and a session state, which consists of key/value pairs. The purpose is to record an extensible set of information regarding the session. Both can in principle been persistent, but currently the session record is used for persistent information, such as the members of a session and what tools they are using, whereas the session state is transient and is used to pass information of a more fleeting nature (e.g. the position of a window) between the participating clients. This is especially used in the tightly coupled session mode, where the tools used by the particpants closely resembles each other. This is quite useful for (among other things) shared navigation through the Web.

Using the session as the central vehicle for collaboration, OHP handles the basic interactions between collaborators. The addition of arbitrary key/value pairs in the session state allows for future extensions. As the session state is broadcast to all collaborators in a session (depending on coupling mode), OHP thus offers finely grained event notification.

One area currently not covered by the OHP is the locking and versioning of documents. The session state could certainly be extended to this purpose, but this would only work within one session. Locking must be handled on higher level to functional appropriately. Another approach would be to add an additional protocol to OHP to handle the locking and versioning of resources. This area has previously been handled in hypermedia systems, and OHP could be extended likewise. As mentioned in Section 3, the need for locking depends on the number and size of the resources to be locked. If resources are small and numerous, the likelihood of collisions is smaller.

The Construct Architecture

Having introduced the Open Hypermedia Protocol, we will now turn to an implementation of the standard. The Construct architecture is a hypermedia server architecture based on the principles of OHSWG. It can be seen in Figure 1. The application layer is in this figure illustrated by the Arakne Environment [7]. The general three-layered architecture compares with the Core Architecture of [25] and many others. The responsibilities of the three layers are:

Application layer The programs that the user interacts with reside here. The application layer is responsible for the presentation of hypermedia structures and the integration with third-party applications, as well as allowing users to interact with others through sessions. The hypermedia structure manipulation is handled by extensible set of hypermedia tools (known as 'Hypermedia Views) through a likewise extensible set of services that address specific kinds of hypermedia, such as navigational or spatial hypermedia.

[1] `http://www.ohswg.org/`

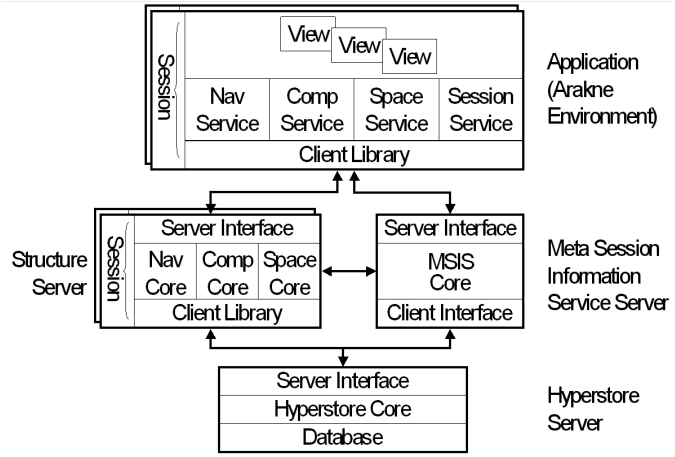


Figure 1: The OHSWG based Construct architecture. The hypermedia tools (or "Views") manipulate the runtime and storage data model through the interaction with the services provided by the Arakne Environment. Through a communication layer the messages from these services are sent to the Structure Server, where the appropriate core processes them, resulting in queries to the Hyperstore Server. Every session has instances of Views as well as services, communication layers, etc. Thus, in a situation with *n* sessions, the above picture would have *n* session boxes within the Application layer and *n* session boxes within the Structure Server.

Structure layer The services above communicate using Open Hypermedia Protocol with the cores found in the Structure Server. These cores are responsible for the actual structural computations (such as in the case of navigational hypermedia 'Follow Link). The cores are also responsible for converting hypermedia structures into generic structures called 'units, which are stored by the storage layer. At this layer we also find the Meta Session Information Server (MSIS), which handles sessions.

Storage layer The Hyperstore stores all hypermedia structures once they have been transformed into units at the structure layer. The current implementation supports two kinds of storage: file-based and an Oracle database.

When comparing this architecture to OHSWG, it should be noted that the OHSWG standard currently only describes the communication between the application layer and the structure layer. Furthermore, this communication is currently limited to navigational hypermedia. The Construct architecture is based on OHSWG and implementation of navigational hypermedia complies with the standard. However, the system has also been extended with support for composites and spatial hypermedia using protocols closely based on the Open Hypermedia Protocol.

The broadcast of messages is, in the Construct implementation, accomplished by giving each session a set of hypermedia cores and communication layers in the structure server. Symmetrically all active sessions at the application layer have their own hypermedia views, hypermedia services, and communication layer.

5 Augmenting the Web with Open Hypermedia

As described in the introduction, the Web is currently unable to handle arbitrary linking and structuring. Several open hypermedia systems have been built to address this, and the Arakne Environment is one such system.

The Arakne Environment is a general environment for an open set of hypermedia tools. Through integration with the Microsoft Internet Explorer, the system currently provides its users with navigational hypermedia with bi-directional multi-headed links in Web pages and in temporal media through Navette [8], as well as the guided tour tool Ariadne [31], and the spatial hypermedia tool CAOS [39].

Using the Arakne Environment and the tools within it, it is possible to create complex structures on top of Web pages. Links and annotations created by the hypermedia views are added, when a Web page is displayed in the user's Web browser, so that it becomes possible to make for instance links to and from pages without modifying the pages themselves (which would require ownership of the pages). For structuring purposes other than navigation, users can create guided tours with Web pages as nodes in the tour, or organise many Web pages using spatial hypermedia. The tools can be employed concurrently, e.g. a user could create links on a Web page that was a part of a guided tour.

Through its support for hypermedia structuring, the Arakne Environment allows users collaborate in the sense, that they can annotate and structure a shared document space. In addition to this basic functionality, the Arakne Environment has been extended to utilise the collaborative aspects of OHP, and how this reflects on functionality as well as user interface is described in the following section.

Handling Collaboration in the Arakne Environment

The current version of the Arakne Environment has been designed to support the collaborative aspects of the Construct server, and thus of the OHSWG standard. This section describes how this manifests itself in practice.

All user interactions with a hypermedia view take place within a session. The user begins by default in a private session, and can then decide to either launch a hypermedia view in that session, create a new session, or to join an existing session. Each session has a runtime data model, shared among the hypermedia views active in the session. Thus, with several active sessions, several runtime data models representing perhaps the same hypermedia structures are present in the Arakne Environment. Rather than a waste of resources, this conforms to the behaviour expected from coupling modes. As described in Section 3, coupling modes dif-

fer in how often updates take place. So, if a user is engaged in two sessions in two different coupling modes, a uncoupled and a tightly coupled session, there should be no ‘spill-over between the sessions, as this would violate the semantics of the coupling modes.

Session states are used to communicate the changes within one user’s hypermedia view to that of other users. A hypermedia view can, depending on its abilities, generate events matching changes in its own state (such as the display of an endpoint, or the movement of a node in a guided tour), and can symmetrically receive such events and update its own state accordingly. The session state consists of key/value pairs and uses a systematic name space to route the updates received to the right hypermedia view.

The foundation for collaboration in the Arakne Environment is the session, and Figure 2 shows a screenshot, where a user is engaged in two sessions. The user is currently working within the session “OHS Session, using the hypermedia views Navette and Ariadne. The Session Manager (shown to the right) is the main interface to interact with sessions and other users. Using the Session Manager users can see, who else is using the system, what views they are using, create new sessions, see what sessions are available, and join (or leave) existing sessions. When a user joins a session, it becomes active, and the user can switch between active sessions through the tabs at the bottom of the Arakne Environment.

The bottom line of the Arakne Environment is the ticker tape. Events and messages can be routed here, if the user so desires. Events scroll by in the ticker tape gradually greying out, as to show their age. Event subscription is handled through the Subscription preferences, seen in Figure 3. The basic event system in the Arakne Environment is based on the messages sent by OHP as well as session state changes. In the figure, the user has selected to be alerted in the ticker tape, whenever the user ’bouvin’ creates or deletes links or nodes in the ’Web research’ session, as well as subscribing to four session state changes. This interface is dynamically created based on the backend’s and the hypermedia view’s capabilities. Depending on the view’s capabilities, it is possible to route notifications directly to the view. The view can then highlight the change reflected by the notification.

6 Collaboration and the Arakne Environment

In this section we will return to the issues raised in Section 3, and investigate Arakne in this context. One of the design goals of the Arakne Environment interface is to support the seamless transition between single-user and collaborative work, and how this is supported will also be discussed.

Event notification

In the Arakne Environment the awareness of others actions are supported in two ways: firstly through the use of sessions, where the awareness is explicit, depending on the coupling mode used, and secondly through event notification.

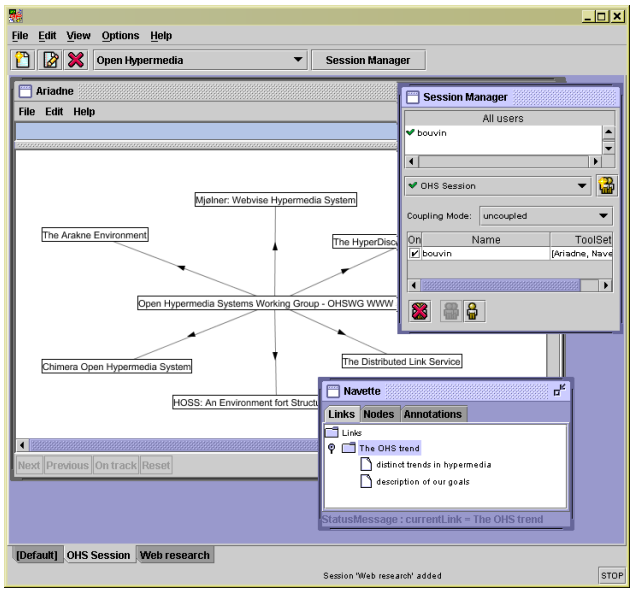


Figure 2: The Arakne Environment. This is is picture of a situation with two active sessions (as seen by the session tabs at the bottom), and two hypermedia views. The Session Manager is shown in the upper right corner

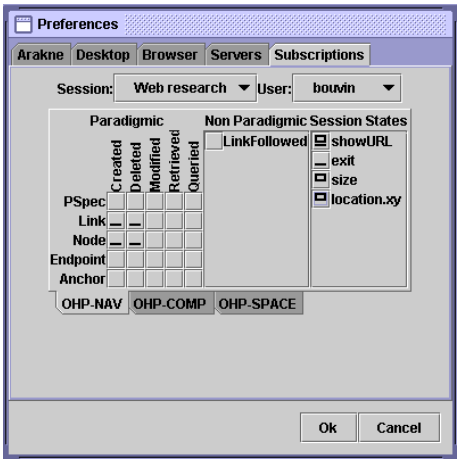


Figure 3: The Subscription interface. Allows a user to subscribe to notifications, and to designate where such notifications should appear. A line signifies the ticker tape, a box in the hypermedia view, and a combination both.

As described above, The Arakne Environment sports a ticker tape at the bottom of the window. The ticker tape is used to display information that can help users stay aware of what other people are doing. Such information includes the creation or modification of hypermedia structures, other actions taken by other users (e.g. link transversal), or users leaving or joining sessions. The messages displayed in the ticker tape is repeated for a set time, determined by the user, and fades during this time, so that a quick glance allows the user to distinguish recent messages from older ones. Messages can have associated actions to them, depending on their type, and the user can select between these actions by clicking on a message. This results in a popup menu with the appropriate actions.

Filtering can be done on the type of event, on the issuer of the event, and on the session the event occurs within. Thus, a user might continue working in session, but setting the subscription such that when another user returned to his or her desk and started working with the Arakne Environment in another session, it would generate a notification in the ticker tape.

The ticker tape provides an relatively non-intrusive mechanism for keeping users peripheral aware of each others' actions. Still the default subscription is no subscriptions at all (apart from being told when other users leave or join the session), so if a user desires no intrusion at all, the system will not force such an intrusion.

It should at this point be noted, that the Arakne Environment until recently did not use a separate channel or protocol for event notification. The notifications received and sent are all based on either OHP or the Session State changes. As described in Section 7, the Arakne Environment is currently being integrated into the iScent event system [2]. The transition to iScent is quite simple (it is undergoing testing as of this writing), and it will enable people not using Arakne to see what is being done, as well as giving the Arakne users an interface to a much more general event notification service.

To provide users with a more informal interface for interaction, the Arakne Environment sports an Internet Relay Chat client. This functionality was originally added to facilitate the logging of debugging information from distributed clients, but it has been trivial to extend this to allow the users to chat with each other. By relying on the IRC standard, it is also possible for users at a computer without the Arakne Environment to interact with the other users by using a standard IRC client. The workplace value of chat tools has been investigated by the BABBLE system [9]. The team behind the development of the Arakne Environment (a distributed team of four people) has used messaging extensively as coordination tool with good results.

Locking
Locking is currently not supported by OHSWG, and thus not by the Arakne Environment. This is a weakness and more work needs to be done in this area. The problem domain of the Arakne Environment is the annotation and structuring of existing Web pages, and the main problem is then the locking of the hypermedia structures themselves. As mentioned in Section 6, this is not necessarily a major problem, especially as users (depending on the coupling mode) are updated continually.

In the context of locking Web documents, Arakne supports WebDAV [41], which is mainly used for publishing annotation as Web pages, but it can also be used to lock other Web documents. As WebDAV becomes widely adopted this functionality will in actual use probably be superseded by the WebDAV locking found in the Web page editors employed by the users.

Persistent collaboration information
As noted above the OHSWG standard provides persistent sessions, so users can resume work within sessions with other users at a later point. User can also resume work in their private sessions, where the state of the views will be restored as they left it. From the standpoint of the interface, a session is just another page with some hypermedia views running in it. It is easy to change between sessions, and it is obvious which sessions are active. The coupling mode is an important part of how session work, and the users can dynamically change the coupling mode of a session, without any other changes in the interface. This allows the users to go smoothly from non-collaborative to collaborative or vice versa.

Versioning
Versioning is a important component of a collaborative hypermedia system, where documents and structures evolve over time. Currently versioning is not a part of the OHSWG specification, and this important functionality is thus missing from the Arakne Environment. The Delta-V working group [14] under the WebDAV project is expected to provide the versioning functionality for documents some time this year, which however still leaves the (harder?) problem of structure versioning. This is further complicated by the extensible nature of the Construct model, which can accommodate many structuring mechanisms. Versioning for navigational hypermedia cannot be expected to be the same as for spatial hypermedia. One solution would be to apply versioning at Hyperstore level, where all structure has been turned into units, but much semantic knowledge might be lost that way.

7 Future Work
Through the development history of the Arakne Environment, it has undergone several informal user testings, mainly involving other team members (not directly developing on the Arakne Environment). This has greatly informed the design, and has led to numerous significant changes. Formal user testings have however not been done yet, but are expected to take place over the summer.

To support more sophisticated event notification mecha-

nisms, the Arakne Environment is currently undergoing a conversion to the iScent event service. iScent (intersubjective collaborative event environment) is designed for general event notification, and has as an explicit goal to support intersubjectivity, e.g. reflected knowledge between coworkers ("I know that you know that I know"). A special feature of the iScent system is the storage of all generated events, which then later can be used to catch up with what has passed. The inclusion of this system is expected to enrich the value of the notification mechanism already present in the Arakne Environment, as the notifications displayed will no longer be limited to "merely" OHP events. In extension of this, the Arakne Environment will be extended with the Trail Viewer, a tool for visualising and manipulating previous actions taken by a user. As all generated events are stored in the iScent system, it becomes an automatic memory extension for the user. Given the number of the events generated, the need for effective structuring and visualisation becomes necessary, and the development of this tools promises to be interesting.

8 Conclusion

Collaborative hypermedia is an exciting field, as is the OHSWG standardisation effort. The session combined with coupling modes offers a strong tool for support for collaborative work, but there still remains much work to be addressed by the OHSWG in this aspect. Most obvious is the current lack of locking and versioning. However, the OHSWG is still a work in progress, and these fields can be expected to receive more attention in the future. The promise of interoperability between heterogeneous hypermedia systems, and collaboration between these systems is grand.

So far the use of awareness tools and notification mechanisms in the context of hypermedia seems promising. Support for collaborative work, while being available, should not intrude or require unnecessary attention or effort to be used, or it will not be used, as argued by Grudin [26]

A simpler and much more widespread system is the Web. While progress is being made with locking and versioning, the hypermedia model underpinning the Web remains fundamentally unsuited for shared structuring and annotation. The combination of open hypermedia and the Web as exemplified in the Arakne Environment enables users to structure the Web in new ways. The Web is supremely well suited for this kind of extension, as it is based on open and well understood document and protocol formats.

With the combination of the open hypermedia structuring and collaboration support, the Arakne Environment enables users to work closely together in the structuring of the Web. The Web is large and not terribly well structured, and the Web augmentation approach allows users to leverage the information found on the Web in new ways.

Acknowledgements

The author would like to thank René Thomsen and Michael Bang Nielsen for their work on version 2.1. Kenneth M. Anderson has been a valuable discussion partner. The author is a member of the Coconut project `http://www.cit.dk/coconut/`, a joint research project consisting of Department of Computer Science, Aarhus University and Tele-Danmark Internet. The Danish National Centre for IT-Research <http://www.cit.dk/> supports the Coconut project.

REFERENCES

[1] R. M. Akscyn, D. L. McCracken, and E. A. Yoder. KMS: A distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–835, July 1988.

[2] K. M. Anderson and N. O. Bouvin. Enabling project awareness and intersubjectivity via hypermedia-enabled event trails. Submitted for publication.

[3] K. M. Anderson, R. N. Taylor, , and E. J. Whitehead, Jr. Chimera: Hypermedia for heterogeneous software development environments. *ACM Transactions on Information Systems*, 18(3), July 2000.

[4] L. Bannon and K. Schmidt. Taking CSCW seriously. *CSCW Journal*, 1(1–2), 1992.

[5] R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkel, J. Trevor, and G. Wötzel. Basic support for co-operative work on the World Wide Web. *International Journal of Human Computer Studies*, 1997.

[6] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollerman. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 1(2), 1992.

[7] N. O. Bouvin. Unifying strategies for Web augmentation. In *Proceedings of the 10^{th} ACM Hypertext Conference*, pages 91–100, Darmstadt, Germany, Feb. 1999.

[8] N. O. Bouvin and R. Schade. Integrating temporal media and open hypermedia on the World Wide Web. In *Proceedings of the 8^{th} International World Wide Web Conference*, pages 375–387, Toronto, Canada, May 1999. W3C.

[9] E. Bradner, W. A. Kellogg, and T. Erickson. The adoption and use of BABBLE: A field study of chat in the workplace. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 139–158, Copenhagen, Denmark, Sept. 1999. Kluwer Academic Publishers.

[10] V. Bush. As we may think. *The Atlantic Monthly*, pages 101–108, July 1945.

[11] L. A. Carr, W. Hall, and S. Hitchcock. Link services or link agents? In *Proceedings of the 9th ACM Hypertext Conference*, pages 113–122, Pittsburgh, USA, 1998.

[12] L. A. Carr, D. D. Roure, W. Hall, and G. Hill. The distributed link service: A tool for publishers, authors and readers. In *Proceedings of the 4th International World Wide Web Conference*, Boston, USA, 1995. W3C.

[13] H. C. Davis, D. E. Millard, S. Reich, N. O. Bouvin, K. Grønbæk, K. M. Anderson, U. K. Wiil, P. J. Nürnberg, and L. Sloth. Interoperability between hypermedia systems: The standardisation work of the OHSWG. In *Proceedings of the 10th ACM Hypertext Conference*, pages 201–202, Darmstadt, Germany, 1999.

[14] Delta-V Working Group. `http://www.webdav.org/wg/#dv`.

[15] P. Dourish and S. Bly. Portholes: Supporting awareness in a distributed work group. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 541–547, May 1992.

[16] D. Engelbart. A conceptual framework for the augmentation of man's intellect. In P. Howerton, editor, *Vistas in Information Handling*, volume 1, pages 1–29. Spartan Books, Washington DC, USA, 1963.

[17] D. Engelbart. Authorship provisions in Augment. In *Proceedings of the IEEE Compcon Conference*, San Francisco, USA, 1984. IEEE.

[18] G. Fitzpatrick, T. Mansfield, S. Kaplan, D. Arnold, T. Phelps, and B. Segall. Augmenting the workaday world with Elvin. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6th European Conference on Computer Supported Cooperative Work*, pages 431–450, Copenhagen, Denmark, Sept. 1999. Kluwer Academic Publishers.

[19] L. Fuchs. AREA: A cross-application notification service for groupware. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6th European Conference on Computer Supported Copperative Work*, pages 61–80. Kluwer Academic Publishers, Sept. 1999.

[20] L. Fuchs, U. Pankoke-Babatz, and W. Prinz. Supporting cooperative awareness with local event mechanisms: The GroupDesk system. In *Proceedings of the 4th European Conference on Computer Supported Cooperative Work*, pages 247–262, Stockholm, Sweden, 1995.

[21] K. Grønbæk, N. O. Bouvin, and L. Sloth. Designing Dexter-based hypermedia services for the World Wide Web. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8th ACM Hypertext Conference*, pages 146–156, Southampton, UK, Apr. 1997.

[22] K. Grønbæk, J. A. Hem, O. L. Madsen, and L. Sloth. Cooperative hypermedia systems: A Dexter-based architecture. *Communications of the ACM*, 37(2):64–74, Feb. 1994.

[23] K. Grønbæk, L. Sloth, and P. Ørbæk. Webvise: browser and proxy support for open hypermedia structuring mechanisms of the World Wide Web. In *Proceedings of the 8th International World Wide Web Conference*, pages 253–267, Toronto, Canada, 1999. W3C.

[24] K. Grønbæk and R. H. Trigg. Design issues for a Dexter-based hypermedia system. *Communications of the ACM*, 37(2):40–49, Feb. 1994.

[25] K. Grønbæk and U. K. Will. Towards a common reference architecture for open hypermedia. *JoDI, Journal of Digital Information*, 1(2), 1997. `http://jodi.ecs.soton.ac.uk/Articles/v01/i02/Gronbak/`.

[26] J. Grudin. Two analyses of CSCW and groupware. Technical Report 323, Department of Computer Science, University of Aarhus, Denmark, July 1990.

[27] J. M. Haake, T. Knopik, and N. Streitz. The SEPIA hypermedia system as part of the POLIKOM telecooperation scenario. In *Proceedings of the 5th ACM Hypertext Conference*, pages 235–237, Seattle, USA, Nov. 1993.

[28] J. M. Haake and B. Wilson. Supporting collaborative writing of hyperdocuments in SEPIA. In *Conference proceedings on Computer-supported cooperative work*, pages 138–146, Toronto, Canada, Nov. 1992.

[29] W. Hall, H. C. Davis, and G. Hutchings. *Rethinking Hypermedia: The MicroCosm Approach*. Kluwer Academic Publishers, Norwell, USA, 1996.

[30] C. Heath and P. Luff. Collaboration and control: Crisis management and multimedia technology in London Underground line control rooms. *CSCW Journal*, 1(1–2):69–94, 1992.

[31] J. Jühne, A. T. Jensen, and K. Grønbæk. Ariadne: A Java-based guided tour system for the World Wide Web. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998. W3C.

[32] P. Kahn, J. M. Nyce, T. Oren, G. Crane, L. C. Smith, R. Trigg, and N. Meyrowitz. From Memex to hypertext: understanding the influence of Vannevar Bush. In *Proceedings of the 3rd ACM Conference on Hypertext*, page 361, San Antonio, USA, Dec. 1991.

[33] C. C. Marshall. Toward an ecology of hypertext annotation. In *Proceedings of the 9th ACM Hypertext Conference*, pages 40–49, Pittsburgh, USA, 1998. ACM.

[34] N. K. Meyrowitz. Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. In *Proceedings of ACM conference on Object Oriented Programming Systems, Languages and Applications (OOPSLA 86)*, 1986.

[35] N. K. Meyrowitz. The missing link: Why we're all doing hypertext wrong. In E. Barrett, editor, *The society of text: Hypertext, hypermedia and the social construction of information*, pages 107–114. MIT Press, Cambridge, USA, 1989.

[36] P. J. Nürnberg, E. R. Schneider, and J. J. Leggett. Designing digital libraries for the hyperliterate age. *Journal of Universal Computer Science*, 2(9):610–622, 1996.

[37] A. Pearl. Sun's link service: A protocol for open linking. In *Proceedings of the 2^{nd} ACM Conference on Hypertext*, pages 137–146, Pittsburgh, USA, Nov. 1989.

[38] W. Prinz. NESSIE: An awareness environment for cooperative settings. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Copperative Work*, pages 391–410. Kluwer Academic Publishers, Sept. 1999.

[39] O. Reinert, D. Bucka-Lassen, C. A. Pedersen, and P. J. Nürnberg. CAOS: A collaborative and open spatial structure service component with incremental spatial parsing. In *Proceedings of the 10^{th} ACM Hypertext Conference*, pages 49–50, Darmstadt, Germany, 1999.

[40] N. A. Streiz, J. M. Haake, J. Hannemann, A. Lemke, W. Schuler, H. Schütt, and M. Thüring. SEPIA: A cooperative hypermedia authoring environment. In *Proceedings of the European Conference on Hypermedia Technology (ECHT 1992)*, pages 11–22, Milan, Italy, 1992.

[41] E. J. Whitehead Jr. and Y. Y. Goland. WebDAV: A network protocol for remote collaborative authoring on the Web. In S. Bødker, M. Kyng, and K. Schmidt, editors, *Proceedings of the 6^{th} European Conference on Computer Supported Cooperative Work*, pages 291–310, Copenhagen, Danmark, 1999. Kluwer Academic Publishers.

[42] U. K. Wiil and J. J. Leggett. Concurrency control in collaborative hypertext systems. In *Proceedings of the ACM Hypertext 1993 Conference*, pages 14–24, Nov. 1993.

[43] U. K. Wiil and J. J. Leggett. The HyperDisco approach to open hypermedia systems. In *Proceedings of the 7^{th} ACM Hypertext Conference*, pages 140–148, Washington DC, USA, Mar. 1996.

[44] U. K. Wiil and J. J. Leggett. Workspaces: The HyperDisco approach to Internet distribution. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proceedings of the 8^{th} ACM Hypertext Conference*, pages 13–23, Southampton, UK, Apr. 1997.

[45] U. K. Wiil and J. J. Nürnberg. Evolving hypermedia middleware services: Lessons and observations. In *Proceedings of the 1999 ACM symposium on Applied computing*, pages 427–436, San Antonio, USA, Feb. 1999.