Proceedings of the 2nd Workshop on Open Hypermedia Systems Hypertext '96, Washington, DC, March 16-17, 1996

Uffe Kock Wiil
Serge Demeyer

UCI-ICS Technical Report 96-10
Department of Information and Computer Science
University of California, Irvine, CA 92717-3425

April 15, 1996

Table of Contents

Workshop Participants

Ajit Bapat	GMD-IPSI, Darmstadt, Germany
Hugh C. Davis	University of Southampton, England
Thomas Dedonno	University of California, San Diego, USA
Serge Demeyer	Brussels Free University, Belgium
Kaj Grønbæk	Aarhus University, Denmark
Jörg M. Haake	GMD-IPSI, Darmstadt, Germany
Wendy Hall	University of Southampton, England
Peter J. Nürnberg	Texas A&M University, USA
Steven E. Poltrock	Boeing Computer Services, USA
Siegfried Reich	University of Linz, Austria
Marc Rittberger	University of Konstanz, Germany
John B. Smith	University of North Carolina, Chapel Hill, USA
Li-Cheng Tai	University of California, San Diego, USA
Richard N. Taylor	University of California, Irvine, USA
Manos Theodorakis	University of Crete, Greece
Randall H. Trigg	Xerox PARC, USA
Yannis Tzitzikas	University of Crete, Greece
E. James Whitehead, Jr.	University of California, Irvine, USA
Uffe Kock Wiil	Aalborg University, Denmark
Kasper Østerbye	Aalborg University, Denmark

Introduction

The 2nd Workshop on Open Hypermedia Systems was held in conjunction with the 1996 ACM Hypertext Conference in Washington, DC[1]. The main objective of the workshop was to provide a forum for exchange of information and discussion of topics for people sharing interests in open hypermedia systems. The specific objectives of the workshop were (1) to provide the latest results from ongoing well-known research projects in the area (e.g., Chimera, DHM, Hyperform, KHS, Microcosm and SP3/4), (2) to allow research topics and new approaches to be presented and discussed, and (3) to work towards common standards and reference models for open hypermedia systems.

The workshop participants were either invited based on previously published papers on open hypermedia systems or selected based on submitted position papers. Position papers were submitted in response to a call for participation. Position papers were used to influence the set of topics to be discussed at the workshop. The 20 participants where equally divided between well-known people and new-comers, which provided an excellent forum for exchange of experiences and ideas, discussion of issues, and presentation of existing and new approaches.

These proceedings contain a brief description of the activities at the workshop as well as (1) a copy of the accepted position papers, (2) a list of the workshop participants, and (3) the final agenda for the workshop. Most of the workshop material is also available from the workshop home page:

http://www.iesd.auc.dk/~kock/OHS-HT96/

We (the organizers) wish to thank the participants for the many inspiring presentations and discussions that made this workshop very successful.

Irvine, California, April 15th, 1996.

Uffe Kock Wiil and Serge Demeyer

---

1 The 1st Workshop on Open Hypermedia Systems was held in conjunction with ECHT '94 in Edinburgh, Scotland.

Proceedings of The 2nd Workshop on Open Hypermedia Systems, Hypertext '96, Washington, DC, March 16-17, 1996.

Workshop Agenda

Saturday March 16th

9:00 Welcome

9:45 Opening Talk: Managing Engineering Information with Hypermedia

- Invited speaker: Steven E. Poltrock

10:45 Coffee Break

11:00 Session 1: Current Status in OHS Projects - Moderator: Serge Demeyer

- Chimera: Richard N. Taylor
- DHM: Kaj Grønbæk
- Hyperform: Uffe Kock Wiil
- KHS: Marc Rittberger
- Microcosm: Wendy Hall
- SP3/4: Peter J. Nürnberg

12:45 Lunch Break

14:15 Session 2: Reference Models for OHS - Moderator: Randall H. Trigg

- Position statement: Serge Demeyer
- Invited statement: Kasper Østerbye
- Invited statement: Kaj Grønbæk

15:45 Coffee Break

16:00 Session 3: Interoperability in OHS - Moderator: Kasper Østerbye

- Position statement: Jörg M. Haake
- Position statement: Hugh C. Davis
- Invited statement: Richard N. Taylor

17:30 End of Sessions

19:00 Informal Dinner

Sunday March 17th

10:00 Session 4: Architectural Support for OHS - Moderator: Jörg M. Haake

- Invited statement: Uffe Kock Wiil
- Position statement: Siegfried Reich
- Position statement: Li-Cheng Tai
- Position statement: Thomas Dedonno
- Position statement: Yannis Tzitzikas

12:30 Lunch Break

14:00 Session 5: OHS and the WWW - Moderator: Wendy Hall

- Position statement: E. James Whitehead, Jr.
- Position statement: Marc Rittberger
- Invited statement: John B. Smith

15:30 Coffee Break

15:45 Session 6: New Directions in OHS - Moderator: Uffe Kock Wiil

- Invited statement: Peter J. Nürnberg
- Invited statement: Ajit Bapat

16:45 Workshop Wrap Up - Future Directions

17:30 End of Workshop

Overview

Open hypermedia systems (OHSs) is an active field of research within the hypermedia community. In the past couple of years, we have seen a rapidly growing interest in design, development and deployment of OHS in various applications areas including digital libraries, computing support for large engineering enterprises, software development environments and education.

Two workshops on OHS were held in 1994, the first in May in Konstanz, Germany [Aßfalg, 1994] and the second in September at ECHT '94 [Wiil & Østerbye, 1994]. The ECHT '94 Workshop on Open Hypermedia Systems was the first in this series of workshops held in conjunction with the ACM Hypertext Conferences. The ECHT '94 workshop gathered several prominent researchers from the field and gave birth to two working groups, one on *interoperability standards* and one on *reference models*. These working groups never fully materialized in terms of starting collaborative efforts in the OHS community, but they did in fact lead to several paper submissions for the Hypertext '96 conference by some of the group members (e.g., [Grønbæk & Trigg, 1996] and [Østerbye & Wiil, 1996]).

The 2nd Workshop on Open Hypermedia Systems gathered 20 people from Europe and the USA (and had to turn down several other interested people). The main objective of the workshop was to provide a forum for exchange of information and discussion of topics for people sharing interests in open hypermedia systems. The specific objectives of the workshop were (1) to provide the latest results from ongoing well-known research projects in the area (e.g., Chimera, DHM, Hyperform, KHS, Microcosm and SP3/4), (2) to allow research topics and new approaches to be presented and discussed, and (3) to work towards common standards and reference models for open hypermedia systems.

The objectives of the workshop overlapped to some extent with the objectives of The Second International Workshop on Incorporating Hypertext Functionality Into Software Systems (HTF II) [Ashman et al., 1996] which also took place in conjunction with Hypertext '96. The HTF II workshop was full as well (20+ participants), which indicates the significant interest in issues relating to hypertext and integration, both in terms of augmenting the human-computer interaction model with hypertext links and in terms of using hypertext links to organize information by integrating existing tools, data formats and information repositories. A joint Birds Of a Feather meeting at the Hypertext '96 Conference was used to exchange information between the participants

of these two workshops and to plan future activities that will bring these two sub-communities closer together.

Three types of topics were taken into consideration when designing the overall agenda of the workshop: (1) the work in progress by some of the major OHS projects, (2) the work by the two working groups started at ECHT '94, and (3) the topics addressed and issues raised by the position papers. The call for position papers suggested the following topics:

* Experiences with integration of third-party applications
* Experiences with the WWW in OHS settings
* Experiences with tailoring OHSs to specific application domains
* Requirements for third-party application developers
* Requirements for the underlying operating system and hardware platform
* Functionality and layers of OHSs
  - run-time support and linking protocols
  - data models and storage
  - system architectures
* Interoperability standards
* Reference models for OHS

The details of the workshop agenda was developed with two things in mind: (1) invited systems should have a chance to describe current status and future plans in their projects and (2) all participants (including those representing invited systems) should be involved actively in the agenda by giving a statement. There were three types of statements:

Opening statements - where the session moderator briefly introduced the topic.
Position statements - focused talks by participants having submitted position papers.
Invited statements - similar to position statements, except that these participants were invited to the workshop based on previous research record.

Opening Session

After a short introduction of all the participants, the workshop started with an invited talk by Steven E. Poltrock entitled *Managing Engineering Information With Hypermedia* [Nelson & Schuler, 1996]. Steve described important requirements (ability to integrate existing third-party tools, support collaborative work, provide version and configuration control, support massive scale, support longevity of documents, and provide integration with the WWW) that OHS must meet to support creation, management, access, and use of richly interrelated systems engineering information. Steve's experiences with Boeing were used as an example throughout the talk.

Session 1: Current Status in Open Hypermedia Systems Projects

The first morning was used to present and discuss the latest results from existing OHS projects. Six different systems were presented in Session 1: the Chimera system from University of California, Irvine [Anderson et al., 1994] by Richard N. Taylor, DHM from Aarhus University [Grønbæk & Trigg, 1992] by Kaj Grønbæk, Hyperform from Aalborg University [Wiil & Leggett, 1992] by Uffe Kock Wiil, KHS from University of Konstanz [Rittberger et al., 1994] by Marc Rittberger, Microcosm from University of Southampton [Davis et al., 1992, 1994] by Wendy Hall, and the SP3/4 system from Texas A&M University [Leggett & Schnase, 1994] by Peter J. Nürnberg.

Session 2: Reference Models for Open Hypermedia Systems

This session featured three statements, a position statement from Serge Demeyer [page 15] and invited statements from Kaj Grønbæk [Grønbæk and Trigg, 1996] and Kasper Østerbye [Østerbye & Wiil, 1996] (these invited statements were based on papers accepted for the *Open Hypermedia* session at Hypertext '96). The work described in all three presentations originate to some extent from the discussions at the ECHT '94 workshop (i.e., the first draft version of the Flag model was presented at the ECHT '94 workshop). The work was actually supposed to take place as a joint effort in the *reference model* working group, but ended up being three different threads all relating to the same topic.

Session 3: Interoperability in Open Hypermedia Systems

This session started with a position statement by Jörg M. Haake [page 25]. The remainder of the session was spent on the first draft proposal for a standard open hypermedia protocol (OHP) presented by Hugh C. Davis [page 27]. This work originates from the working group on *interoperability* that was started at ECHT '94. Hugh's presentation was followed by a response on the OHP by Richard N. Taylor. The working group started on the OHP just months before the workshop and the OHP as described in these proceedings is still at a very early stage. The intention behind the draft proposal was to start a discussion on the elements of a standard protocol for communication between applications and OHS.

Session 4: Architectural Support for Open Hypermedia Systems

This session featured one invited statement (based on a paper accepted for the *Open Hypermedia* session at Hypertext '96) by Uffe Kock Wiil [Wiil & Leggett, 1996] and position statements by Siegfried Reich [page 55], Li-Cheng Tai [page 61], Thomas Dedonno [page 67] and Yannis Tzitzikas [page 75]. Unfortunately, four of the

participants were missing in this session, since there was an overlap between tutorials and workshops at the conference. Nevertheless, this session featured some very productive discussion and suggestions. This session fostered the idea that the OHS community needs a number of scenarios that can be used (1) to define the scope and application domains of OHS by example and (2) to compare the capabilities of existing OHS in terms of the types of scenarios they support.

Session 5: Open Hypermedia Systems and the WWW

The only moderator that used the opportunity to make an opening statement was Wendy Hall. Wendy described the WWW interoperability activities in the Microcosm project. Wendy's statement was followed by position statements by E. James Whitehead, Jr. [page 81] and Marc Rittberger [page 87]. John B. Smith concluded the session with an invited statement describing the activities at UNC to merge the capabilities of the DGS system [Shackelford et al., 1993] with the WWW. This project is described in [Dewan et al., 1995] and on the WWW (**http://www.cs.unc.edu/~jbs/research/www-dgs/overview.html**).

This session was remarkable in showing that many researchers in the hypermedia community finds it difficult to deal with the success of the WWW. The general feeling is that the WWW community neglects many important lessons that the hypermedia community learned the hard way and that the WWW community is now to a large extent reinventing the wheel. The question then is how the hypermedia community should try to influence the future development of the WWW. A range of possibilities came up such as using Mosaic and Netscape as third-party browsers in OHSs for demonstration purposes, development of enhanced WWW browsers (and editors), submitting papers to WWW conferences, and getting into the program committees of WWW conferences.

Session 6: New Directions in Open Hypermedia Systems

This session featured two invited statements each describing the latest research of a major hypermedia group. These statements were based on papers accepted for the *Systems and Infrastructures* session at Hypertext '96. Peter J. Nürnberg started with a description of HOSS (Hypermedia Operating System Services) [Nürnberg et al., 1996]. Ajit Bapat described the HyperStorM engine, which is the latest work in the area of hypermedia platforms at GMD-IPSI [Bapat et al., 1996].

Closing Session

The discussions at the Closing Session focused on two topics: (1) forming new working groups based on the discussion at the workshop and (2) discussing the need for and the agenda of a 3rd Workshop on Open Hypermedia Systems at Hypertext '97. With respect to the former, the workshop participants decided to start one new working group addressing two key issues: *defining open hypermedia systems* and *defining open link services*. The *OHS Working Group* will be divided into two subgroups. The *scenarios* group will address the issue of *defining open hypermedia systems* and the *protocol* group will address the issue of *defining open link services*. The following people agreed to act as organizers of the work in the two subgroups:

Scenarios:

Peter J. Nürnberg, Texas A&M University.

Jörg M. Haake, GMD-IPSI, Darmstadt

Protocol:

Hugh C. Davis, University of Southampton

Antoine Rizk, EUROCLID

Uffe Kock Wiil, Aalborg University

Since the workshop a home page (maintained by Peter J. Nürnberg) has been created for the *OHS Working Group* (**http://www.csdl.tamu.edu/ohs/**). The introduction on the home page further specifies the initial mission of the two subgroups (the following text has been edited to match the presentation style of the proceedings):

OHS Working Group Home Page

Purpose of the Group. The OHS Working Group formed out of the 2nd Workshop on Open Hypermedia Systems, held March 16-17, 1996, in Washington DC, in conjunction with the ACM Hypertext '96 conference. Its initial mission is to address two key issues: *defining open hypermedia systems* and *defining open link services*.

Defining Open Hypermedia Systems. The decision was made to try to define the scope of open hypermedia systems (OHSs) by example. That is, instead of defining the systems themselves, we want to construct a *canonical* set of *scenarios* that describe problems that we feel are addressed by OHSs. These scenarios should give the hypermedia research community a common terminology that can be used to measure the relative strengths and weaknesses (and the relative applicability) of different approaches to our problem domain.

Defining Open Link Services. The effort to define open link services is embodied in the emerging definition of a standard link services *protocol*. Through the protocol, various levels of hypermedia awareness on the part of the application, as well as various levels of functionality on the part of link servers, are implicitly defined. We expect that this protocol will not only address the research question of the nature of open link services, but also provide a practical method of allowing interoperability between different OHSs.

Why This Is Really the Work of One Group, not Two. On the surface, the above two aims may seem independent. However, we feel that the construction of the scenarios must inform the design of the protocol. That is, the protocol cannot be designed without grounding in actual examples of how we envision open hypermedia systems being used in the real world to solve real problems.

The present work in the *OHS Working Group* and the whole process towards OHS definitions, standards and reference models is open. Thus, we invite all interested parties to join the work. Details on how to join can be found on the working group home page. Contributions to the work can be done at different levels ranging from making comments on the work in progress to providing complete, detailed scenarios of using OHS in specific application domains.

The workshop participants also decided to have a 3rd Workshop on Open Hypermedia Systems in conjunction with Hypertext '97 (Southampton, England, April 6-11, 1997). The agenda of the 3rd workshop will to a large extend reflect the ongoing work in the *OHS Working Group*. Uffe Kock Wiil agreed to organize the next workshop.

References

Anderson, K. M., Taylor, R. N. & Whitehead, E. J. Chimera: Hypertext for Heterogeneous Software Environments. *ECHT '94 Proceedings*, pp. 94-107.

Ashman, H., Balasubramanian, V., Bieber, M. & Oinas-Kukkonen, H. (eds.). *Proceedings of The Second International Workshop on Incorporating Hypertext Functionality Into Software Systems*, (Washington, DC, March), 1996.

Aßfalg, R. (ed.). *Proceedings of the Workshop on Open Hypertext Systems*, (Konstanz, Germany, May), 1994.

Bapat, A., Wasch, J., Aberer, K. & Haake, J. M. HyperStorM: An Extensible Object-Oriented Hypermedia Engine. *Hypertext '96 Proceedings*, pp. 203-214.

Davis, H. C., Knight, S. & Hall, W. Light Hypermedia Services: A Study of Third Party Application Integration. *ECHT '94 Proceedings*, pp. 41-50.

Davis, H. C., Hall, W., Heath, I., Hill, G. & Wilkins, R. Towards An Integrated Information Environment With Open Hypermedia Systems. *ECHT '92 Proceedings*, pp. 181-190.

Dewan, P., Jeffay, K., Smith, J. B., Stotts, D. & Oliver, W. Early Prototypes of the Repository for Patterned Injury Data. *Digital Libraries '95 Proceedings*, pp. 123-130.

Grønbæk, K. & Trigg, R. H. Toward a Dexter-based Model for Open Hypermedia: Unifying Embedded References and Link Objects. *Hypertext '96 Proceedings*, pp. 149-160.

Grønbæk, K. & Trigg, R. H. . Design Issues for a Dexter-based Hypermedia System. *ECHT '92 Proceedings*, pp. 191-200. Also in *Communications of the ACM*, , 37, 2 (Feb.), 1994, 40-49.

Leggett, J. J. & Schnase, J. L. Viewing Dexter with Open Eyes. *Communications of the ACM*, 37, 2 (Feb.), 1994, 76-86.

Nelson, P. R. & Schuler, D. Managing Engineering Information With Hypermedia. 1996. Available from Steven E. Poltrock (poltrock@atc.boeing.com).

Nürnberg, P. J., Leggett, J. J., Schneider, E. R. & Schnase, J. L. Hypermedia Operating Systems: A New Paradigm for Computing. *Hypertext '96 Proceedings*, pp. 194-202.

Rittberger, M., Hammwöhner, R., Aßfalg, R. & Kuhlen, R. A Homogeneous Interaction Platform for Navigation and Search in and from Open Hypertext Systems. *RIAO '94 Proceedings*, pp. 649-663.

Shackelford, D. E., Smith, J. B. & Smith, F. D. The Architecture and Implementation of a Distributed Hypermedia Storage System. *Hypertext '93 Proceedings*, pp. 1-13.

Wiil, U. K. & Leggett, J. J. The HyperDisco Approach to Open Hypermedia Systems. *Hypertext '96 Proceedings*, pp. 140-148.

Wiil, U. K. & Leggett, J. J. Hyperform: Using Extensibility to Develop Dynamic, Open and Distributed Hypertext Systems. *ECHT '92 Proceedings*, pp. 251-261.

Wiil, U. K. & Østerbye, K. (eds.) *Proceedings of the ECHT '94 Workshop on Open Hypermedia Systems*, (Edinburgh, Scotland, September), 1994. Department of Computer Science , Technical Report R-94-2038, Aalborg University. Oct. 1994.

Østerbye, K. & Wiil, U. K. The Flag Taxonomy of Open Hypermedia Systems. *Hypertext '96 Proceedings*, pp. 129-139.

The Zypher Meta Object Protocol

POSITION STATEMENT SUBMITTED TO
THE 2ND WORKSHOP ON OPEN HYPERMEDIA SYSTEMS
Hypertext '96 Conference - March '96 - Washington (US)
http://www.iesd.auc.dk/~kock/OHS-HT96/ - http://www.acm.org/siglink/ht96/

Serge Demeyer
Patrick Steyaert - Koen De Hondt - Wim Codenie - Roel Wuyts - Theo D'Hondt
Vrije Universiteit Brussel / Faculty of Sciences
Programming Technology Lab (PROG) Pleinlaan 2
B-1050 Brussels (Belgium)
phone: (+32) 2 629 34 91
{sademeye | prsteyae | kdehondt | wcodenie | tjdhondt}@vnet3.vub.ac.be; rwuyts@is1.vub.ac.be
http://progwww.vub.ac.be/

Abstract

This paper discusses the necessity of a meta object protocol in the design of an open hypermedia system. It shows that a meta object protocol enables to tailor the behaviour and configuration of the hypermedia system, independent of its constituting elements.

The approach is demonstrated by means of the Zypher Open Hypermedia Framework, where the meta object protocol eases the incorporation of system services (i.e. caching, logging, authority control and integrity control) and flexible reconfiguration (i.e. run-time extensibility and cross-platform portability).

1. Introduction

To understand the argumentation unfolded in the main body of the paper, it is necessary to emphasise that Zypher[1] was explicitly designed as an open hypermedia system with three levels of tailorability. Each level provides different facilities to suit the behaviour of the system to the needs of particular hypermedia applications.

Domain Level

Domain level tailorability aims to deliver hypermedia systems for a specific problem domain by extending the basic hypermedia framework with domain specific modules. Creating such domain specific modules requires a great deal of technical expertise about the software systems applied in the problem domain but has little to do with the hypermedia system as such. One doesn't need to understand the inner details of the hypermedia system to tailor the system. Note that some modules, if written 'good', can be reused for different problem domains.

Typical usage of domain level tailorability is the incorporation of modules for special viewer applications (i.e. Microsoft Word, a HTML browser), extra storage devices (i.e. the local file system, a HTTP-server) and designated navigation facilities (i.e. special URL resolution algorithms).

System Level

System level tailorability aims to deliver services that affect the global behaviour of the hypermedia system itself and requires some knowledge about the internal architecture of the hypermedia system. Services attained trough system level tailorability can be applied on different incarnations of the hypermedia framework: once we have implemented the technique in one framework incarnation, it requires little effort to reuse the code in other incarnations.

Typical examples of services that can be accomplished with system level tailorability are things like logging (maintaining a log of certain activities in order to provide backtracking features), authority control (check whether the user of the system has the privileges to perform certain operations), caching (predict future behaviour on the basis of registered activities) and integrity control (control operations in order to preserve the consistency of the system's data structures).

---

1 The name Zypher stems from the Louis Zypher character performed by Robert De Niro in the movie "Angel Heart".

Configuration Level

Configuration level tailorability aims to provide a 'plug and play' hypermedia system, where the system configuration is adapted without modifying the constituting modules. This accommodates for a flexible system set-up, where new modules can be installed easily. Configuration level tailorability requires a deep knowledge about the internal architecture of the hypermedia system; however technical details about individual modules do not matter.

Typical examples of configuration level achievements are flexible configuration (i.e. run-time extensions to the system) and portability (i.e. cross-platform reconfiguration).

These levels of tailorability are quite important and hypermedia system designers will often need to switch between these levels in order to develop a particular hypermedia application. That is why we have devised special icons that are employed in the framework documentation and throughout the remainder of this text[2]. The icons are based on the *puppet master metaphor* (see [figure 1]).

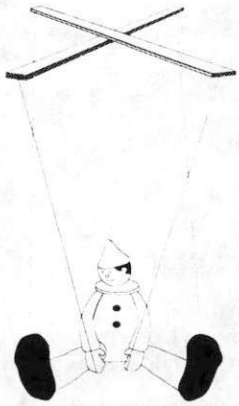


Figure 1: Puppet Master Metaphor

- When preparing a story, the puppet designer will conceive a number of puppets playing different characters. To distinguish these characters the puppets will be dressed with different costumes and their faces will be painted. Typical puppet characters are the harlequin and the pierrot, the former wearing a costume with lots of coloured patchwork and a smiling face, the latter is dressed in white with a tear under the eye. This kind of tailorability corresponds with the domain level tailorability and is visualised using an icon presenting a puppet.
- However, for certain kinds of stories, some puppets require special abilities that demand for extra strings to manipulate the special behaviour. Some scene in the play might benefit from a horse with a swinging tail, in which case the puppet designer will take an existing horse puppet and attach a new string to the tail. A puppet designer that attaches new strings to puppets is a designer that operates on the system level of tailorability, which is denoted by means of a hand-with-string icon.
- Finally, the way the strings work together is implemented in the wooden cross manipulated by the puppet player. A puppet designer creating a knight on a horse fighting with a spear will adapt the branches of the wooden cross to operate the puppet and works on the configuration level of tailorability. This is symbolised with a cross icon.

(*remark:* Note that the users of the hypermedia system correspond with the audience watching the puppet: they are not supposed to know how the puppet is manipulated to produce the desired scenes in the performance. However, just like the audience can influence the play by applauding and shouting, users can influence the behaviour of the system by setting preferences. Actually, the puppet master —i.e. the hypermedia system designer— will use the appropriate level of tailorability to satisfy the audience).

Document Organisation

The remainder of this paper will be used to demonstrate how techniques from the object-oriented software engineering community may help to develop and maintain a hypermedia system with the three levels of tailorability. More precisely, the aim of the paper is to show that the introduction of a *meta object protocol* delivers the desired system level and configuration level tailorability (see sections 3 and 4). Before discussing the notion of a meta object protocol, we will discuss the issue of domain level tailorability (see section 2), where we will define an *object-oriented framework* for the domain of Open Hypermedia Systems.

2. The Base Level

This section will settle the scope of the rest of the paper, with a design specification for the Zypher Open Hypermedia Framework. It is important to note that this specification is not complete and this for three reasons. First of all, the design will be gradually improved (by adding meta objects) when we introduce the notions of system level and configuration level tailorability. Secondly, because the design of the base level is not crucially important for the real issue: the necessity of a meta object protocol). All that really matters is that there exists a base level design, and that it is formalised in a set of contracts specifying relations between objects (we refer the interested reader to [Demeyer'96] for a full specification in design pattern form). Thirdly —and this follows from the previous motive— because it is possible to introduce a meta object protocol on any system with a base level design based on an object-oriented framework. The third point is extremely important, as it makes the technique of a meta object protocol applicable in many other hypermedia systems.

---

2 The idea of visualising the levels of the system by means of icons is adapted from [Kiczalis,Rivières,Bobrow'91].

A Design Specification

The design of the Zypher framework was based on the Dexter model [Halasz,Schwartz'90], well known in the Hypermedia community. Zypher retained the separation between the storage layer and the run-time layer (called presentation layer in Zypher) and the main elements of the Dexter factorisation (i.e. component, anchor, instantiation and marker). To handle the specific problems of an 'Open' hypermedia model[3], we extended the model with elements that represent a viewer application (i.e. editor), an information repository (i.e. loader) and a link resolution algorithm (i.e. a resolver) . This resulted in the object model (using OMT notation; see [Blaha,Premerlandi,Rumbaugh'88] and [RumbaugEtAl'91]) depicted in [figure 2].

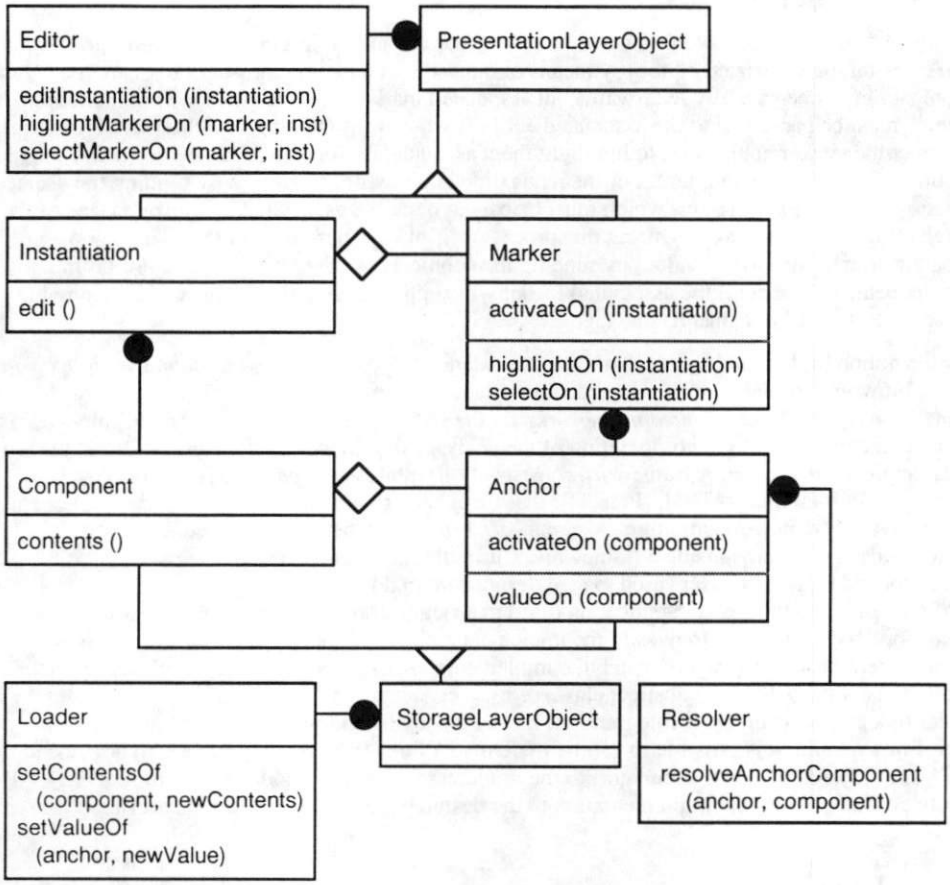


Figure 2: The Design of the Base Level

Instead of presenting an explicit enumeration of all contracts, that exist between the objects defined in [figure 2], we will give a brief description of the message flow that implements the navigation operation (the heart of all hypermedia models).

Step 1: Selection of navigation source

An instantiation represents a document as it is displayed by some viewer application (the viewer application is represented by the editor object). An instantiation contains a number of markers (representations for the visible sources or targets for navigation operations) which explains the aggregation relation between instantiation and marker.

To start a navigation action, the editor will send an #activateOn message to a marker with the associated instantiation as parameter. The marker is allowed to produce some visual effects before proceeding with the next step.

Step 2: Identification of navigation source

The 1-to-many associations between instantiation/component and marker/anchor will be used to find the associated anchor-component pair. This pair identifies the source of the navigation operation and the marker must send the #activateOn message on to the associated anchor supplying as parameter the component associated with the instantiation.

---

3 See [Demeyer'96] for a motivation of these extensions.

Step 3: Resolution Process

The 1-to-many association between anchor/resolver will be used to retrieve the resolver object containing the algorithm that produces the target of the navigation action. The #resolveAnchorComponent message must be send to this resolver to obtain a collection of quadruples where each quadruple represents one target of a navigation action. Inspired by the Dexter model, each quadruple contains a component specifier, a component presentation specifier, an anchor specifier and an anchor presentation specifier; such quadruple will be turned into a new component, anchor, instantiation and marker that will serve as the target of the navigation operation.

Step 4: Target Presentation

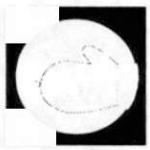
For each target the message #edit will be sent to the instantiation; the instantiation must pass this message on the associated editor (by means of the #editInstantiation message) to instruct the viewer application to open a view. Afterwards, all associated markers will be sent the #highlightOn message which must be passed on to the associated editor (by means of the #highlightMarkerOn message) to instruct the viewer application to highlight them as candidate sources or targets for future navigation actions. Finally, the actual target of the navigation action will be selected by sending the #selectOn message to the target marker, which must be passed on to the associated editor (by means of the #selectMarkerOn message). During this process, it is always possible to request a component for its contents (an anchor for its value) by sending the #contents (#valueOn) message. For un-initialised components (or anchors) the associated loader will supply the actual contents (value) using the #setContentsOf (#setValueOf) message.

To verify the notion of domain level tailorability, we extended Zypher with several modules for the so-called "framework browser" problem domain. In order to explain the framework browser concept, we must clarify some related concepts. *Object-oriented frameworks* are the state of the art in object-oriented software engineering and consist of a tight co-operation of the analysis, design and implementation concepts modelling a particular application domain. A framework consists of different *design patterns* (see [Johnson'92], [Beck,Johnson'94], [GammaEtAl'93], [Pree'94]) that focus on a single analysis, design and implementation aspect of the overall framework structure. A *framework browser* is then an integrated set of tools to manipulate the design patterns inside a framework. Currently, these tools are

- a home cooked HTML browser (used to read design pattern documentation),
- the Microsoft Word third party application (used to produce design pattern documentation),
- several code browsers (used to modify the implementation of the framework) and
- several pattern browsers (used to match the implementation expressed in concrete classes with the design specified in contracts between abstract classes).

The Zypher link engine seamlessly integrates all these tools by providing navigation facilities from one tool to another. For example, it is possible to follow hypermedia links from the design pattern documentation (i.e. a HTML or Microsoft Word document) to the implementation (i.e. a code or pattern browser). Also, one can make hyper jumps from the implementation to the design pattern documentation describing that part of the framework.

3. The Meta Level



To verify the notion of system level tailorability, we decided to experiment with a backtrack function for all the navigation actions performed by the hypermedia system. A backtrack function is often helpful in hypermedia systems, as it is one of the techniques to handle the well known 'lost in hyper space' phenomenon (i.e. [Zellweger'89] and [Conklin'87]).

Keeping track of all navigation actions boils down to the maintenance of a log: for all navigation actions we must save the internal state of the participating agents to be able to restore them later. From this insight follows that an implementation must solve two problems in order to provide a working backtracking service. There is the problem how to ensure that all navigation actions are witnessed and there is the problem how to save the internal state of the participants. The former problem will be discussed in the following section; the latter is beyond the scope of this paper. Briefly we can say that the introduction of special state objects memorising the internal state of an object solves the problem. The technique is based on the 'Memento' design pattern, as described in [GammaEtAl'93]; we refer the interested reader to [Demeyer'96] for a more detailed description.

Funnel Navigation Actions

Re-examining the base level design of the Zypher system (see [figure 2]), we find that the navigation operation is modelled with a few key messages defined on the participating agents (i.e. the #activateOn message defined on marker, the #activateOn message defined on anchor, or the #resolveAnchorComponent defined on resolver). If we want to log all navigation actions, this would imply a patch of all implementations of at least one of these key messages. From a software engineering perspective, this is an unfavourable situation as it causes redundancy: the implementation of the logging algorithm is duplicated

over all implementations of the patched key message. From an open hypermedia perspective the situation is even worse, because in an extensible set-up, the objects participating in the navigation action may be supplied by external sources. This means that there is no secure way to incorporate the log algorithm in all implementations, which implies that we can not ensure the integrity of the log.

To ensure that all navigation operations are witnessed by the log algorithm we must adapt the design of the hypermedia system by providing a funnelling point for all navigation actions. In the Zypher design, such funnelling point is accomplished in the so-called 'path'[4], an object with the explicit responsibility to control all navigation operations. The adapted model is depicted in [figure 3] (to avoid a cluttered figure, we left out some of the objects and most of the messages).

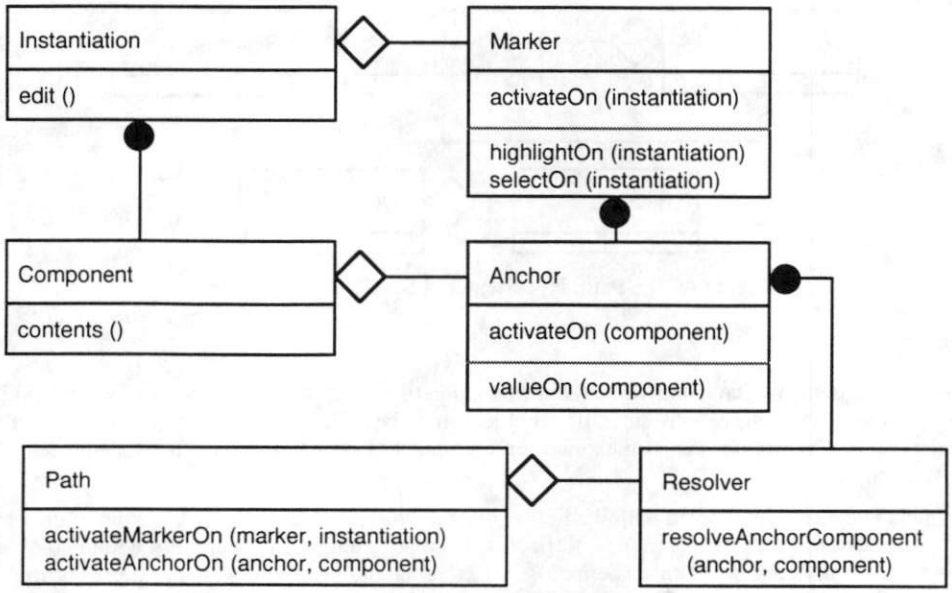


Figure 3: The Path Meta Object

The semantics of the adapted model (see [figure 3]) is as follows. There is exactly one path object for each hypermedia system. The implementation of the #activateOn message on all marker objects must delegate to the global path object implements by means of the #activateMarkerOn message; the implementation will perform the four steps involved in the navigation operation (i.e. selection of navigation source - identification of navigation source - resolution process - target presentation) by sending the appropriate messages to the participating objects.

Funnel Storage & Presentation Layer operations

Services like authority control, caching and integrity control have much in common with the logging example from above. They all depend on the ability to control all occurrences of particular messages being sent, regardless of the objects involved. For example, to implement authority control one wants to adapt all implementations of all #edit messages on all instantiations to check whether the user has the appropriate privileges; to maintain a cache of visited information one wants to patch all implementations of all #contents, #setContents, #valueOn and #setValueOf messages on all components and anchors; to ensure the integrity of the system's data structures one will control all operations that change the associations between the objects.

Like argued above, the best way to control all occurrences of a particular message is to provide a funnelling point. The design of the Zypher framework includes the 'session' object to funnel all presentation layer operations and the 'hypertext' object to funnel all storage layer operations[5]. The result is depicted in [figure 4].

---

4 The name stems from the work of Zellweger [Zellweger'89].

5 The Dexter model [Halasz,Schwartz'90] furnished the names 'hypertext' and 'session' because these objects are responsible for the management of the storage and presentation layer respectively.

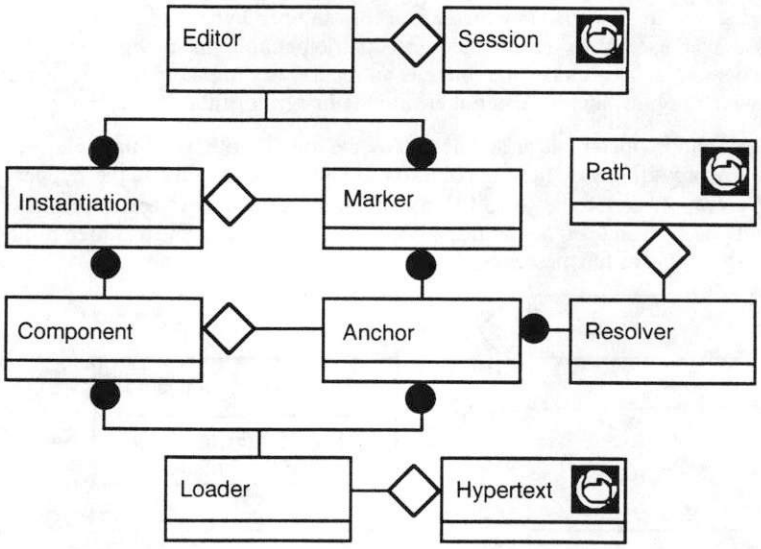


Figure 4: The Path, Hypertext and Session Meta Objects

Why Meta ?

In the previous sections, we have motivated the introduction of 'funnel' objects to provide system level tailorability. Now, we will argue why such funnel objects may be called meta objects. The argumentation relies on the fact that a meta objects is an explicit representation of contracts defined between objects in the base level design.

The term meta is generally connoted with the notion of *reflection*, i.e. the ability of a system to inspect and modify representations of it's own activities. Reflection is an intriguing idea —certainly within computer science— but is mostly considered an academic issue. Reflection has been studied in the area of artificial intelligence and the design of computer languages for quite a long time now (i.e. [Maes'87], [Kiczalis,Rivières,Bobrow'91], [Steyaert'94]). There, it has been shown that reflection eases extensibility (i.e. define a small and fixed kernel language and use that kernel to extend the language expressiveness), backward compatibility (i.e. compatibility with older definitions of the language) and efficiency (i.e. differ the implementation strategy to optimise behaviour). Moreover, since a reflective system is able to monitor its own activities, powerful tools like debuggers and code optimisers can be constructed more comfortably.

Recently the idea has been applied on the design of systems other than programming languages (i.e. [Rao'91]), leading to what has been called *implementational reflection* (or sometimes *open implementations*). A system with implementational reflection has the ability to inspect and/or change the implementational structures of its subsystems. Implementational reflection does not directly provide solutions for the problem domain the system has been designed for, but it does contribute to the internal organisation and the external interface of that system. This suggests that what we have been calling system level tailorability is indeed a feature that can be attained with implementational reflection.

To explain why the funnel objects make the hypermedia system a reflective one, we turn to the definitions found in [Maes'87]. There, *a reflective system is defined as a system which is about itself in a causally connected way*. We elaborate on the three main ideas in this definition (i.e. system, about-ness and causal connection) to make things more precise. A '*system*' is software running on a computer with the intention to answer questions about and/or support actions in some domain. A system will incorporate internal structures representing it's domain, that is why a system is said to be '*about*' it's domain. A system is said to be '*causally connected*' to its domain if the internal structures and the domain they represent are linked in such a way that if one of them changes, this leads to a corresponding effect on the other. In an object-oriented implementation of a system, the parts of the system that represent causally connected internal structures are called meta objects.

The definition of causal connection implies that a causally connected system may actually cause changes in the problem domain by a mere change in the internal representation of that problem domain. As a consequence (since a reflective system incorporates structures that are causally connected to itself) a reflective system can actually modify itself by changing its internal representation.

To argue why the funnel objects (i.e. path, session, hypertext; see [figure 3] and [figure 4]) defined in the previous sections are meta objects, we must prove that these objects are (a) about the hypermedia system in (b) a causally connected way. The proof follows from the insight that the funnel objects are *explicit representations of the contracts defined between the objects on the base level*. Indeed, the important messages are specified in the static part of the contracts (i.e. the interface of the different objects as shown in [figure 2]),

thus part of the design. However, without the funnel objects, the dynamic part of the contracts (i.e. the decision when a certain message is sent) is delegated to the implementation and it is precisely the dynamic part of the contracts that determines the system's behaviour. If the design is extended with the specially created path, session and hypertext objects (see [figure 3] and [figure 4]) the dynamic parts of the contracts are explicitly available, since all occurrences of all important messages arrive at, or originate from such funnel objects. Knowing that the specially created path, session and hypertext are representations of the dynamic parts of the contracts between the base level objects —specifying how the system should behave under certain conditions— , they are by definition 'about' the system. Moreover, they are an explicit representation of the contracts: changing the implementation of a funnel object will have immediate effect on the subsequent behaviour of the system so we conclude that they are 'causally connected' to the system.

4. The Meta Meta Level

In the previous section we have introduced meta objects (i.e. path, session and hypertext) as the explicit representations of the contracts defined between base level objects (i.e. component, anchor, instantiation, marker, resolver, editor and loader). However, the introduction of meta objects leads to supplementary contracts, which raises the question whether it is worthwhile to make these supplementary contracts explicit as well.

To show that it is worthwhile, this section will start with a summary of the contracts introduced by the meta level objects, followed by a description of two experiments with configuration level tailorability and ending with a discussion on the connection between meta meta objects and configuration level tailorability.

Meta Object Contracts

The meta objects are defined as objects controlling all operations concerning a particular layer (i.e. path for the navigation layer, session for the presentation layer and hypertext for the storage layer). As we can expect from this definition, there are more operations defined on meta objects, as the ones that follow from funnelling base level operations. A quick look at the design with the meta level objects (see [figure 4]) learns that the introduction of the meta objects adds operations for the aggregation relationships path-resolver, session-editor and hypertext-loader. The role of these aggregation relationships is to specify what kind of resolvers, loaders and editors are installed in the hypermedia system, which corresponds to the management of the available peripheral systems of the hypermedia system. Likewise (not visible in [figure 4]) the meta objects participate in aggregation relationships specifying the available classes (i.e. hypertext - component class; hypertext - anchor class; session - instantiation class; session marker class), which corresponds to the supervision of the potential elements constituting the running hypermedia system. Finally, (not depicted in [figure 4]), the meta objects introduce operations to create, query and release associations between objects (storage layer object - presentation layer object in session; instantiation - marker and component - anchor in path) plus relations between normal objects and peripheral objects (presentation layer object - editor in session; storage layer object - loader in hypertext; anchor - resolver in path). The role of these operations is to govern the connections between the internal elements of the hypermedia system and the link with the outside world.

The previous paragraph is a very short description of the contracts introduced by the meta level objects (for a detailed description we refer to [Demeyer'96]) which shows that —besides funnelling the navigation, presentation and storage layer operations— the meta objects do implement the configuration of the hypermedia system. This suggests that an explicit representation of the contracts defined on meta level objects may lead to the required configuration level tailorability.

Configuration Level Tailorability Experiments

To explore the notion of configuration level tailorability, we conducted two experiments. The first one is based on an interpretation of the URL (universal resource locator) format for anchors as defined in the HTML specification (see [Berners-LeeEtAl'94]). HTML documents embed their anchors in their documents using the URL format and we applied the same technique for the Microsoft Word documents. The URL format is open in the sense that it is prefixed by a keyword identifying the target address space, followed by an address in a format depending on the keyword prefix. The list of possible keywords is in principle unlimited, so the linking potential is only bounded by the list of interpretable keywords. In the Zypher hypermedia framework, the link engine consists of the list of resolvers installed in a path, so the mapping of the keyword-prefix on the appropriate resolver is the crucial process in the configuration of the hypermedia system. This mapping process is available by means of the #determineResolverFor message on the path meta object.

The second experiment has to do with the system's configuration across platforms. The Zypher framework documentation is organised as a collection of design patterns containing embedded anchors referring to related design patterns. The referencing is done by name, i.e. there is a special URL format starting with the keyword 'pattern' and followed by the name of the particular design pattern. The pattern resolver will turn this name into a file-name containing the design pattern document. However, the design pattern may come in a HTML and Microsoft Word version. The Microsoft Word version is richer (i.e. can be edited, contains pictures) but is

only available on the Windows platform. On other platforms software engineers must use the HTML version. Moreover, some design patterns documents are 'read-only' and may only be opened with a HTML browser. The above conditions influence the decision on what configuration of component, instantiation, anchor and marker objects to use for the representation of the target document. In the Zypher hypermedia framework, such decisions are implemented in the processes that interpret the specifier-quadruples returned by the resolver and turn them into actual component, anchor, instantiation and marker objects. These processes correspond with the #interpretComponentSpec, #interpretAnchorSpec, #interpretComponentAndPSpec, #interpretAnchorAndPSpec messages defined on the path object.

Configuration and Meta Meta objects

The messages #determineResolverFor, #interpretComponentSpec, #interpretAnchorSpec, #interpretComponentAndPSpec, #interpretAnchorAndPSpec defined on the path meta object are part of the navigation layer contract defined on the meta level. As argued in the case of the base level objects ([figure 2]), the mere presence of these messages in the design means that only the static part of the contracts is explicitly available which is not enough to control the execution of the contracts. To attain configuration level tailorability, we must make the dynamic parts of the contracts explicit, which is precisely the role of the hypermedia context object ([figure 5]).

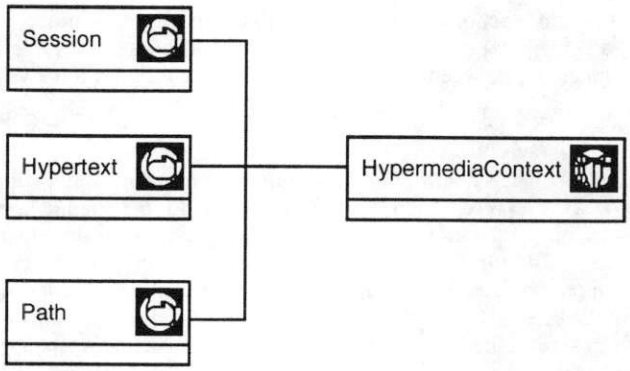


Figure 5: The Hypermedia Context

The implementation of the #determineResolverFor message on the path object must request the HypermediaContext object to return the name of the resolver that must be used (by means of the #determineResolverFor message defined on HypermediaContext). Likewise, the implementation of the #interpretComponentSpec, #interpretAnchorSpec, #interpretComponentAndPSpec, #interpretAnchorAndPSpec must request the HypermediaContext for the name of the classes that should be instantiated (by means of the #determineComponentClassFor, #determineAnchorClassFor, #determineInstantiationClassFor, #determineMarkerClassFor messages).

Just like the meta objects (session, hypertext and path) funnelled all operations dealing with one aspect of the hypermedia system (i.e. presentation, storage and navigation), the HypermediaContext meta meta object funnels all operations controlling the configuration of the hypermedia system. To change the subsequent configuration of base level objects, only one object must be modified.

Note that the parameters passed to, and the results returned from, the messages sent to the HypermediaContext object are always (collections of) strings. This ensures that the system's configuration never depends on particularities of base level objects, which accounts for the 'plug and play' requirement in system level tailorability. Moreover, it allows to implement the configuration messages as look-up tables which are very easy to maintain, even by end users (i.e. compare with the table of 'helper applications' maintained by most World-Wide-Web browsers).

5. Conclusion

One of the ideas that came up during the previous open hypermedia workshop (i.e. [Wiil,Østerbye'94]) was to develop some kind of an 'open hypermedia reference model', similar to what the Dexter model [Halasz,Schwartz'90] did for the generation of monolithic hypermedia systems. The idea behind this paper is to show that, if such an attempt is to succeed, the reference model should embody the notion of system level tailorability (i.e. the incorporation of system services like caching, logging, authority control and integrity control) and configuration level tailorability (i.e. flexible configuration of the system to support run-time extensibility and cross-platform portability). This paper confirms that —although the notions of system and configuration level tailorability may seem quite complex to implement— the technique of a meta object protocol brings it within range of today's software engineering.

The introduction of a meta object protocol has been defined as a process with the following steps. (a) Develop a design specification of an object-oriented framework for an open hypermedia system. Such a specification

defines contracts between objects representing the main elements in the design. (b) Define meta objects as an explicit representation of contracts between objects in the base level design. Define a meta object protocol as the protocol specifying how base level objects exchange messages with meta objects. (c) Define a meta meta object as the explicit representation of contracts between objects in the meta level and classes in the base level. Define a meta meta object protocol to establish associations between objects in the base level.

The three levels of tailorability match nicely with the stepwise introduction of the meta object protocol. I.e. step (a) corresponds with domain level tailorability, step (b) accounts for system level tailorability, and step (c) enables configuration level tailorability. What is particularly appealing in the light of an 'Open Hypermedia Reference Model' is the fact that —since steps (b) and (c) are quite independent from the particular design resulting from step (a)— the meta object approach is also applicable to other open hypermedia systems implemented with an object-oriented framework.

6. References

[Beck,Johnson'94] Beck, K. / Johnson, R. "Patterns Generate Architecture"; ECOOP'94 Proceedings, Lecture Notes in Computer Science nr. 821, Springer-Verlag, 1994. Check http://st-www.cs.uiuc.edu /users /patterns /patterns.html for anonymous ftp.

[Berners-LeeEtAl'94] Berners-Lee / Cailliau, R. / Luotonen, A. / Nielsen, H. F. / Secret, A. "The World-Wide Web"; Communications of the ACM - Vol. 37(8) - August '94.

[Blaha,Premerlandi,Rumbaugh'88] Blaha, M. R. / Premerlandi, W. J. / Rumbaugh, J. E. "Relational Database Design Using an Object-Oriented Methodology"; Communications of the ACM - Vol. 31(4) - April '88.

[Conklin'87] Conklin, J. "Hypertext: An Introduction and Survey"; IEEE Computer - Vol. 20 (9) - September 1987.

[Demeyer'96] Demeyer, S. "Zypher: A Hypermedia System Incarnated In a Framework Browser"; Phd. dissertation, forthcoming. Check http://progwww.vub.ac.be/zypher/.

[GammaEtAl'93] Gamma, E. / Helm R. / Johnson R. / Vlissides, J. "Design Patterns: Abstraction and Reuse in Object-Oriented Designs"; ECOOP'93 Proceedings, Lecture Notes in Computer Science nr. 707, Springer-Verlag, 1993. The same people have written the book "Design Patterns"; Addisson-Wesley, 1995.

[Halasz,Schwartz'90] Halasz, F. / Schwartz, M. "The Dexter Hypertext Reference Model"; Proceedings of the 1990 NIST Hypertext Standardisation Workshop (January 16-18, Gaithersburg, MD). Republished in Communications of the ACM - Vol. 37(2) - February '94.

[Johnson'92] Johnson, R. "Documenting Frameworks Using Patterns"; OOPSLA'92 Proceedings, ACM Press, 1992. Check http://st-www.cs.uiuc.edu /users /patterns /patterns.html for anonymous ftp.

[Kiczalis,Rivières,Bobrow'91] Kiczalis, G. / Rivières, J. / Bobrow, D. G. "The Art of the Metaobject Protocol"; MIT Press, 1991.

[Maes'87] Maes, Pattie "Concepts and Experiments in Computational Reflection"; OOPSLA'87 Proceedings, ACM Press, 1987.

[Pree'94] Pree, W. "Design Patterns for Object-Oriented Software Development"; Addisson-Wesley 1994.

[Rao'91] Rao, R. "Implementational Reflection in Silica"; ECOOP'91 Proceedings, Lecture Notes in Computer Science, P. America (Ed.), Springer-Verlag, 1991.

[RumbaugEtAl'91] Rumbaugh, J. Blaha, M. / Premerlandi, W. / Eddy, F. / Lorenson, W. "Object-Oriented Modeling and Design"; Prentice Hall, 1991.

[Steyaert'94] Steyaert, P. "Open Design of Object-Oriented Languages"; Phd. dissertation, Vrije Universiteit Brussel, 1994. Check http://progwww.vub.ac.be/prog/papers/paperquery.

[Wiil,Østerbye'94] Wiil, U. K. / Østerbye, K (editors) "Proceedings of the ECHT'94 Workshop on Open Hypermedia Systems"; Technical report R-94-2038 / Institute for Electronic Systems Department of Mathematics and Computer Science - Fredrik bajers Vej 7 - DK 9220 Aalborg - Denmark. Check ftp://ftp.iesd.auc.dk/pub/packages/hypertext/ECHT94-workshop/.

[Zellweger'89] Zellweger, P. T. "Scripted Documents: A Hypermedia Path Mechanism"; Hypertext'89 Proceedings, ACM Press, 1992.

The role of application integration in open hypermedia systems

(Position paper for the 2nd Workshop on Open Hypermedia Systems at Hypertext '96)

Ajit Bapat, Jörg M. Haake

GMD-IPSI
Dolivostr. 15
D-64293 Darmstadt, Germany
Tel.: ++49-6151-869960
Fax: ++49-6151-869966
e-mail: {bapat,haake}@darmstadt.gmd.de
WWW: http://www.darmstadt.gmd.de/~{bapat,haake}

1. Introduction

One aspect of open hypermedia systems is the ability to integrate external applications into a hypermedia system. More often than not, the pieces of information whose relationships to one another are maintained by the hypermedia system are created and manipulated outside the hypermedia system using external (i.e., legacy) applications.

In open hypermedia systems specific problems related to the heterogeneous distributed nature of the environment arise:

- Which application is to be used and how is it invoked?
- How can the consistency of documents be enhanced/supported?
- How can external applications deal with link information in hypermedia objects?
- How to store and provide the hypermedia data?

2. Identification and invocation of external applications

In the case of open hypermedia systems interoperating across heterogeneous networks information about an external application's location, working directory, calling syntax, etc. can no longer be maintained locally like, e.g., a mailcap-file as used by WWW browsers. The information may vary from (network) node to node and applications existing on one node may not exist on another.

An approach to overcome this problem is to provide an application integration service (AIS) that is known in the whole network. Instead of calling an external application directly, the open hypermedia system requests the AIS to execute the desired application(s). Mapping mechanisms within the AIS will ensure the correct invocation of the application. By providing a set of abstract application classes the AIS overcomes the problem of platform-specific applications and can even support user preferences for applications. The AIS will also manage any transport of data that is necessary.

3. Consistency of data

Another major issue that arises when integrating external applications is the consistency of data: How to

deal with any manipulation of data that occurs outside the control of the open hypermedia system? This can be a problem if, e.g., link information of a document is maintained by the OHS while the document itself is manipulated by the external application.

One approach could be to store the document "within" the OHS (see section 5), i.e., external applications can only access the data if the application was called via the open hypermedia system (e.g., using the AIS).

A less rigid approach would be to provide means to the open hypermedia system that allow it to recognize that a manipulation has taken place outside the OHS.

4. Link support

Handling hypermedia links when using external applications is a further issue. There are two mechanisms to maintain links: either embed the anchors within the hypermedia object or maintain a seperate link server.

In the case of embedded links the external application has to be (made) hypermedia-aware, i.e. it must know how to display and manipulate the contained link information. For hypermedia-unaware external applications an intermediate "filter" between the OHS and the application might be a solution.

In the case of a separate link server a hypermedia-aware application could notify the link server about any changes affecting the link information. For hypermedia-unaware application, again, some intermediate layer would have to be added.

5. Providing the data

The hypermedia data that is to be manipulated by the external application can be maintained (and provided) by the open hypermedia system in different ways. One way would be to keep it in a centralized repository (as, e.g., in SP4) and only extract it for the time the external application needs access. This would --- as mentioned above --- eliminate the problem of data consistency.

Another way would be to have a distributed repository like the different WWW servers all over the world resemble.

A third way could be a decoupled storage where data is stored independently from the open hypermedai system. One example might be a document management system. Access to the storage system would be provided by the AIS which would perform any necessary data transport from and to the storage system before resp. after the invocation of the external application.

6. Conclusions

We have argued that the introduction of an application integration service (AIS) is a promising approach to a number of issues that arise when integrating external applications into open hypermedia systems. Others, like the issue of link maintainance, are to be tackled next.

OHP:
A Draft Proposal for a Standard Open Hypermedia Protocol (Levels 0 and 1: Revision 1.2 - 13th March. 1996)

Hugh Davis, Andy Lewis
The Multimedia Research Group
Electronics and Computer Science
The University of Southampton
Southampton
UK, SO17 1BJ
email hcd@ecs.soton.ac.uk , ajl@ecs.soton.ac.uk

Antoine Rizk
Euroclid
email Antoine.Rizk@inria.fr

Abstract

This paper describes a proposal for a protocol known as OHP, for communication between applications and hypermedia link services. If hypermedia viewers were written to use this protocol, or third party applications were adapted to use the protocol, then these applications could be used with any link service which adhered to the protocol. The paper proposes the use of shims for converting between the OHP protocol and the linkservice's native protocol.

Contents

1. Introduction

In order to introduce this paper we would first like to define what we mean by "open" when referring to hypermedia systems. A number of authors (Davis et al., 1992; Østerbye & Wiil, 1996) and workshops (Aßfalg, 1994) (Wiil & Østerbye, 1994), have attempted to define the term "open hypermedia", and we believe that the following is a reasonable summary of current thinking.

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

1. *Size*: It should be possible to import new nodes, links, anchors and other hypermedia objects of any limitation, to the size of the objects or to the maximum number of such objects that the system may contain, being imposed by the hypermedia system.
2. *Data Formats*: The system should allow the import and use of any data format, including temporal media.
3. *Applications*: The system should allow any application to access the link service in order to participate in the hypermedia functionality.
4. *Data Models:* The hypermedia system should not impose a single view of what constitutes a hypermedia data model, but should be configurable and extensible so that new hypermedia data models may be incorporated. It should thus be possible to interoperate with external hypermedia systems, and to exchange data with external systems.
5. *Platforms*: It should be possible to implement the system on multiple distributed platforms.
6. *Users*: The system must support multiple users, and allow each user to maintain their own private view of the objects in the system.

In this paper we are concerned with enabling applications to make them link service aware, so that users may have access to the full range of hypermedia functionality from their standard desktop environment. Many of the current generation of hypermedia systems such as DHM (Grønbæk & Trigg, 1992) HyperDisco (Wiil, 1996), Microcosm (Davis et al., 1994), Multicard (Rizk & Sauter, 1992) and the Texas A&M system prototypes (e.g. Kacmar & Leggett) have addressed this problem, but so far no standard has emerged due to the different hypertext data models and communication protocols adopted by these systems.

Unfortunately, linkservice protocols tend to be kept confidential, and are anyway too detailed to form the topic of published papers. However, we have now seen the results of sufficient research in this subject to be able to identify the common themes, and hopefully to abstract the common requirements. This first draft of this protocol leans heavily upon the M2000 protocol (Rizk, 1991) developed for use with Multicard, and the Microcosm message model. It is a first attempt at such a standard protocol, and no doubt it will need adapting and will evolve as system designers attempt to map their systems onto the protocol.

We should emphasise at this point that all this protocol attempts to provide is a standard method for applications, such as word processors, graphics viewers and drawing programs, to communicate with a link service so that they may offer hypermedia services to their users. This protocol is not an attempt to produce a complete standard for all inter-component communication, such as would be required for the link service to communicate with agents and other service providing components.

2. The Protocol Shim

In general, open hypermedia systems tend to implement a *link service layer* which provides the storage and access mechanism for links and anchors. The Dexter reference model (Halasz & Mayer, 1994) refers to this as the *storage layer*. Some systems call this layer a *hyperbase*; such systems usually store not only the links and anchors, but also the nodes in this layer. The distinctions between the different types of system are ill defined, and anyway not the subject of this paper. The thing that all such systems have in common is that the application layer (or runtime layer) is physically separate from the storage layer, and that there is some system of message passing between the two, which allows the applications (editors, viewers) to send and receive messages and query the link service concerning hypertext services.

All systems designers have developed their own private protocols for communicating with the link service, and these protocols are so deeply buried into the system code that in general it would involve a major re-implementation to rewrite the system to adhere to some new, standard protocol. However, the topics of these communication are essentially similar and it should quite possible to abstract a common set of messages. We suggest that the difference between different system protocols could be resolved if each system produced a *protocol shim* which would sit between the application and the link service as shown in figure 1.

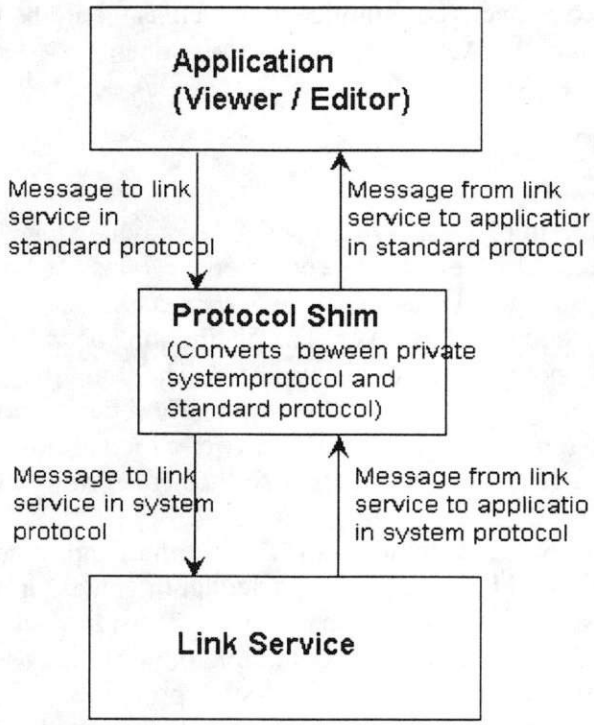


Figure 1: Protocol conversion using a protocol shim.

The advantages of this scheme are:

- Once an application (for example a word processor) had been adapted to communicate using the standard protocol (for example, using an application macro language) , then it would be possible to circulate the code extensions or macros so that any system would be able to benefit from this "hypertext aware" application.
- Each system developer need develop just one extension to their system (the protocol shim itself)
- Applications that have been developed specifically as hypermedia viewer/editors could be used in other systems, so long as they observed the protocols.

The problem with this approach is that some hypermedia systems have much heavier weight requirements on their protocols than others. However many developers have recently acknowledged that it is necessary to accept the concept of "levels of awareness" (e.g. Davis et al., 1994, Wiil, 1996), and that it will not always be possible or reasonable to expect the highest level of hypertext functionality from every third party application. Reflecting these pragmatic observations we propose that the OHP protocol will have a number of levels of conformity, and it would thus be possible to describe a given application as OHP aware to some given level.

3. Anchors

One of the major differences between hypertext implementations is the way that anchors have been treated (Davis, 1995). The issues, from the point of view of hypertext enabling third party applications, are:

- where anchors are stored;
- how new anchors are allocated;
- how the anchor refers to the object within the node data that is to be the physical manifestation of that anchor.

Questions about the behaviour of anchors (are anchors actually processes, and what happens when an anchor is activated?) are not of interest in this context, as they are dealt with within the link service rather than within the application. For the purposes of this discussion we will follow Leggett & Schnase's (1994) distinction between an *anchor* (a hypertext object which describes one or other end of a link) and a *persistent selection* (that object within the node data which is the physical manifestation of the link anchor, such as a coloured text string).

Probably the most generic description of the anchor is provided by Grønbæk & Trigg (1996), in which an anchor is described as a binding between a component identifier and a *location specifier* (LocSpec). A LocSpec contains the details which describe a persistent selection, and the semantics of these details are decided by the application at the time that the anchor is created and resolved using the same algorithm when the anchor is loaded again at a later time. A similar generalisation of the nature of these details comes from the HyTime standard (DeRose & Durand, 1994) which allows that anchors might reference objects within the node using:

- *counting:* e.g. "the string at bytes 1023-1043 through the file", "the 8th line of the 12th paragraph", "the rectangle of dimensions (45,20) pixels at co-ordinates (1427,2393) in the bitmap", etc.;
- *naming*: e.g. "the bookmark called 'MyBookmark' ", "the spreadsheet cell range called 'Total' ", "the drawing object named 'GasMain4' ", "the section of a text named 'Conclusions' ", etc.;
- *searching:* e.g. "the first occurrence in the file of the string 'Xanadu' ", "the record in a database that results from this SQL query", etc.

HyTime also allows that references might be made of combinations of these methods, e.g. "The first occurrence of the string 'Xanadu' in the 5th paragraph of the chapter named 'history' ".

Different systems allocate anchors in different ways. There are four principle methods:

1. The application embeds the anchor in the node data. In this case the application may also embed all the link information at this point (e.g. URL's embedded in html as HREF's) or might allocate an ID to the anchor which is unique to this node, and which will specify to the link service the link or links in which the anchor participates, as in Hyper-G (Andrews et al., 1995).
2. The application allocates an ID for the anchor, which will be unique for that node, and takes responsibility for maintaining a table of IDs and LocSpecs belonging to the node. The link service can resolve an anchor by using the (node name, anchor ID) pair. This is the approach taken by most Dexter based systems such as DHM.
3. The application requests that the link service allocates an ID for the anchor, which will be unique

over the entire link service. The application will still need to maintain a table of anchor IDs and LocSpecs at run time. This approach is taken by Multicard and the hyperbase class of systems.

4. The application talks to the linkservice by transmitting the node identifier and LocSpec, and allocates no specific anchor ID at all. This is the approach taken by Microcosm.

In designing our protocol, we must produce a standard which allows existing and future hypermedia systems to participate, regardless of the method they use to represent anchors or LocSpecs or the way that they allocate and store anchors.

So far, the discussion of anchors has assumed that all anchors will be represented by some persistent selection or button within the node data when the data is displayed. Hypertext links are invoked by clicking on the button. This traditional interface to hypermedia is well established, but has limitations that are discussed in Hall (1994). An alternative interface to hypermedia is the "selection and action" metaphor which has been pioneered in Microcosm, which enables the user to make dynamic queries of the system, for example by selecting some text string and asking the system to compute links using some dynamic information retrieval algorithm. The protocols will need to support this form of hypertext interaction.

4. Communication Protocols

The question of the choice of communication protocol is interesting. The communication is clearly peer-to-peer in the sense that either the shim or the node viewer might initiate a message. We could agree that any suitable method could be used (e.g. DDE on Windows, AppleEvents on the Mac, RPC on Unix), but this approach has problems:

- applications that have been enabled for one communication protocol will only be transferable to other systems if that other system has a shim using that protocol.
- it would be preferable to build shims that will allow clients on one machine architecture to communicate with link services on other architectures.

For these reasons it makes sense to separate the choice of communication protocol from the hypertext protocol, as shown in figure 2. In this configuration, the application developer produces a communication shim to run on the client side. This shim understands the application protocol (e.g. DDE) and converts it to whatever network protocol is required (e.g. sockets). As we shall see later, this shim needs to be more than just a communication protocol converter, and is required to have some hypertext functionality. The linkservice developer produces the protocol shim, which runs on the same platform as the linkservice, and accepts incoming messages in whatever network protocols are supported.

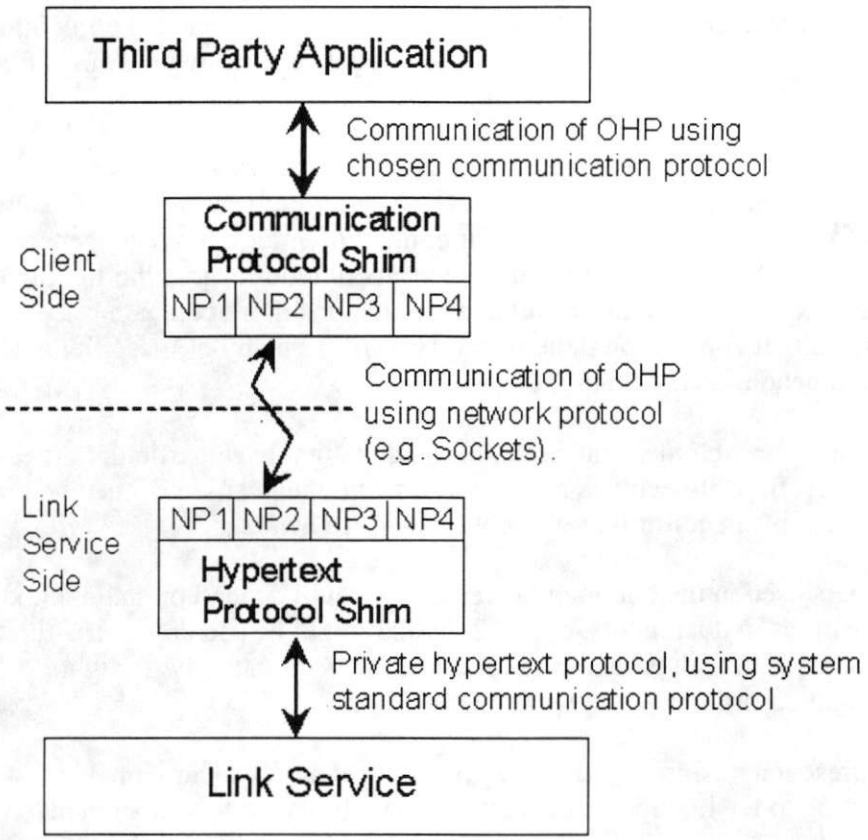


Figure 2: Communication Protocol Conversion

5. Defining the OHP Protocol

5.1. LocSpecs

The difficulty in defining a standard LocSpec is that if it is to be sufficiently flexible to meet all possible ways of expressing positions in a document as intended by the HyTime standard (see section 3) then it will be difficult to parse and thus put considerable onus on the application. Instead we suggest the following simplification, at least for the purpose of getting the standard started. We expect that it will evolve.

```
LocSpec ::=

{\ContentType MimeType
\Content Mime encoded text string
\Count [Comma separated list of numbers]
\ReverseCount [Comma separated list of numbers]
\Name [Mime encoded text string]
\Script [Viewer Executable Script]}
```

The important thing to realise about the LocSpec is that although the link service may be required to store this information, it is not usually required to interpret the information in any way. The semantics of

the various fields within the LocSpec are decided by the viewer at the time that it creates the LocSpec, and then stored in the linkservice for later retrieval and interpretation by the same viewer. Thus different viewers are not required to keep to any particular convention for representing these fields.

The one exception to the above rule is the content. Some hypermedia systems store the content of the anchors within the linkservice, and use this information as the basis for providing services. For example, Microcosm searches its linkbases for links authored on particular content in order to provide generic links. For this reason, this part of the LocSpec is not optional. The protocol requires that the LocSpec contains at least one method for the viewer to store information about the position of an anchor. However, it is strongly recommended that application developers try to fill as many of these fields as possible. The reasons for this recommendation are that:

- some link services may attempt to search their databases for anchors that have particular attributes;
- extra information is generally useful if it becomes necessary to mend the position of the anchor after a document has been edited by an editor that was not linkservice aware.

The comma separated list of numbers used in the count and reverse count will depend on the method used by the viewer to represent positions in documents, e.g. 3, 2, 5, 423 might be used by a structured document to mean the 423rd character in the 5th paragraph of section 3.2. Alternatively it might represent the top left and bottom right co-ordinates of a rectangle in a bitmap.

The reverse offset allows us to represent a position by counting from the end of a document. This overloading can be useful when trying to fix LocSpecs that have become broken due to document editing.

The name may be any object understood by the viewer which is unique to the current document, such as a bookmark name or a drawing object.

The script is a set of instructions which may be interpreted by the application which will enable the application to locate the object.

5.2. Presentation Specifics

Many systems incorporate the possibility of displaying persistent selections in different ways. For example, persistent selections in text might be coloured blue, or some other colour, and areas of bitmaps might be shaded, raised or outlined; some systems support buttons that are not shown, or are only shown when the pointer is over them. In general this is a user choice, and it applies to all the persistent selections in the view, so may be a setting on the viewer that is completely independent of the link service.

However, there are cases where an author may wish particular persistent selections to show in some way that is different from others. In this case it is possible for the viewer to allow the specification of some particular presentation of the selection, and to store this along with the LocSpec, so that when it is viewed later, it will have the desired appearance. If this were the only requirement, the syntax and semantics of the presentation specifier would belong only to the viewer.

Unfortunately, there are many cases where the link service may wish, as the result of some script, to change the presentation, and the linkservice will not know the presentation unless it is part of the

standard.

For the purpose of this version of the protocol we suggest that the information about presentation is kept viewer specific. i.e. the viewer may ask to store any presentation information in any form that it chooses, and the viewer will interpret this data at a later stage when asked to present the data again.

The consequence of this policy will be that link services will not be able to communicate to the viewer that it should change any presentation (e.g. as the result of a script) since it will not know the syntax of the presentation tag for this particular viewer. Later versions of the protocol should standardise the form of this information, e.g. by defining the data as {colour, style, visibility} and giving a set of values and meanings to each of these parameters.

5.3. Scripts

Scripts play an important part in some hypermedia systems (e.g. Hypercard and Multicard) , and are hardly used in others. Scripts may be classified into two types:

Server End Scripts. These are the scripts that are carried out when some particular event occurs or some action is requested. From the point of view of the viewer there is no difference between some process being run or a link being followed by the linkserver. Of course, these scripts may cause new messages to be sent back to the viewer, to change its presentation in some way.

Client End Scripts. These are scripts that are sent to the viewer by the linkserver. In general they are either sent as part of a LocSpec in order to identify an anchor, or they are sent as a process which the viewer will be expected to run, for example in order to change the presentation of the data in some way. OHP must provide support for such scripts if the viewer wishes to use them.

5.4. Communication Channels

It is quite possible that one communication shim may need to maintain communication with more than one application on a client machine, and also quite possible that one application might be concurrently displaying more than one node (e.g. an MDI application in Windows). Similarly, one link server (and protocol shim) may need to maintain communication with more than one client. For this reason it is necessary that messages contain routing information, so that messages sent by the link service are sent to the correct node in the correct application on the correct machine. However, from the point of view of the application developer, it is not necessary to know anything about the content of the routing information. The link service will provide the information to the application, and the application will be responsible for returning this information every time it sends a message to the link service, so that the link service can identify where to reply. This information is known in the protocol as the **Channel**.

Since the linkservice will maintain a unique channel to each document that is currently connected, it will not be necessary for most of the messages to pass the document name, as this is implicit in the channel. However, there would be nothing to prevent the linkservice explicitly using the document name as part of the channel information.

5.5. Form of the Protocol

OHP is a peer to peer asynchronous protocol. Messages may be sent by either the link service or by the editor / viewer, and no message waits for any form of reply. If a reply is expected, it is up to the receiving component to initiate a new message. There are thus two classes of message to be considered; those sent by the viewer and those sent by the link service.

Messages that the viewer may send will depend largely on the set of services that can be handled by the link server. It therefore makes sense that part of the protocol will allow the definition of the subjects which the link server can handle. These subjects do not need to be known in advance.

On the other hand, the messages which the link server may send to the viewer must be defined in advance, since the viewer must be coded to deal with these messages.

OHP messages will consist of text strings. In the following sections the messages are shown formatted on multiple lines for ease of presentation. However, the messages themselves will be one continuous stream of ASCII text, unbroken by line breaks. The messages consist of Tags, which are proceeded by a backslash ('\') and succeeded by a space. The characters that follow, up to the next tag or the end of the message are the tag contents. A tag content may be empty if shown as optional in the message definition. The tags of each message are all compulsory, and must be presented in the order in the message definition. If a backslash ('\') or closing curly bracket ('{' or '}') occurs within the tag content, then it should be quoted by preceding it with a further backslash. The bold '+' is used in the following descriptions to indicate that one or more of the line indicated may be included at this point.

We can now begin to define the protocol.

5.6. Messages that the linkserver may send

5.6.1. HeresServices

```
\Subject HeresServices
\Data {\Menuitem Menuitem\Service Service}+
\Channel Channel
```

Which will define the menu that will be placed on the viewer, and the set of services that will be requested from the link service when such items are selected from the menu.

For example, a Microcosm link server would typically send the following:

```
\Subject HeresServices
\Data {\Menuitem Follow Link\Service FOLLOW.LINK}
\Data {\Menuitem Show Links\Service SHOW.LINKS}
\Data {\Menuitem Compute Links\Service COMPUTE.LINKS}
\Data {\Menuitem Make Start Anchor\Service START.LINK}
\Data {\Menuitem Make End Anchor\Service END.LINK}
\Channel #9
```

5.6.2. LaunchDocument

```
\Subject LaunchDocument
\DocumentName DocName
\ReadOnly True/False
```

```
\DocumentNickName [DocNickName]
\DocumentType MimeType
\DataCallBack True or False
\Channel Channel
```

This message introduces a special problem. At the time that it is sent there may be no viewer open to receive the message! The linkserver has requested that a particular document of a particular type is opened, but at present the viewer for this data type is not running. It is necessary that there is some component at the client end to receive this message that can arrange to fire up the appropriate viewer. We suggest that this functionality should be added to the communication protocol shim, which is anyway at the client end. The shim will need to know which viewer to load, depending upon the mime type of the data. The viewer, when it has been launched will receive this message will be responsible for storing the channel information and using it whenever sending messages back to the linkservice.

A further problem is introduced by the fact that some systems would expect the client viewer to now arrange to fetch its own data (using the document name to describe a file on the file system or a remote server) whereas other systems might expect that the data will be held within the linkservice (hyperbase). The DataCallback tag, allows the linkservice to indicate that it wishes the viewer to send a message back asking for the data, and if such a call-back occurs it will answer with a HeresDocument message. If the call-back is false the linkservice will expect the viewer to arrange to load its own data.

It is also a useful point to introduce the idea of levels of the protocol. Many linkservices allow third party applications to participate in the hypertext, in so far as they may be launched and run with given node data, and they may be closed. This is the functionality that is supported by OHP Level 0. In this case the will be expected to handle messages to open the node (LaunchDocument) and close the node (CloseNode). The application itself is not expected to deliver and hypermedia functionality. All other messages in this document are part of the Level 1 Protocol. Level 2 of the protocol is not yet defined, but is discussed in the section 8. The client end communication shim is responsible for knowing the level of the applications it launches so that it does not attempt to send messages to non-communicating applications, but instead returns error messages. In this respect the shim acts as a proxy for the application.

The ReadOnly Tag enables the link service to inform the viewer that it should not allow the user to edit the node contents.

The DocumentNickName allows for the case where the linkservice may supply better descriptions of the document than the filename alone, and the viewer may elect to display this name somewhere, typically in the title bar.

5.6.3. HeresDocument

This message is sent by the linkservice if it has received a call-back asking the linkservice to supply the document data.

```
\Subject HeresDocument
\Data Mime Encoded Data
\Channel Channel
```

5.6.4. HeresAnchorTable

After a document is launched the viewer may request the anchor table. This linkservice will reply with this message.

```
\Subject HeresAnchorTable
\Data {\AnchorID AnchorID
\LocSpec LocSpec
\Direction Start/End/Bidirect
\Presentation colour,style,display
\Service Service}+
\Channel Channel
```

An anchor record is thus seen to consist of an unique identifier, a LocSpec, a direction and a service.

The direction field allows the linkservice to specify whether the given anchor is the start of a link, the end of a link, or one end of a bi-directional link.

The service field allows for the case that not all hypertext buttons will require a Follow Link type of service. Sometimes a anchor may be created that causes some other service (e.g. dynamically generate links) to be requested.

5.6.5. DisplayAnchor

When a document is launched, or possibly at some other time, it will often be necessary to send a message requesting that a particular anchor is displayed, e.g. by moving the cursor to the anchor, or highlighting the anchor. This message is sent to the viewer to request such an action.

```
\Subject DisplayAnchor
\AnchorID AnchorID
\Presentation colour,style,display
\Channel Channel
```

5.6.6. DisplayLocSpec

Sometimes it is necessary to display something other than a pre-defined anchor, for example, the result of a search. This message may define the object to be viewed.

```
\Subject DisplayLocSpec
\LocSpec LocSpec
\Presentation colour,style,display
\Channel Channel
```

5.6.7. HeresNewAnchor

The protocol requires that the viewer is not responsible for allocating anchor ID's. Instead a message will be sent to the linkservice asking for an anchor ID, and the linkservice will reply:

```
\Subject HeresNewAnchor
\Data {\AnchorID AnchorID
\LocSpec LocSpec
\Direction Start/End/Bidirect
\Presentation colour,style,display
\Service Service}
```

```
\Channel Channel
```

5.6.8. Interpret

This message allows the link service to send scripts to the application which the application will interpret. Such scripts might alter the presentation of a node, change the contents of a node or anything else that the interpreter contained in the application is capable of undertaking.

```
\Subject Interpret
\ScriptType ScriptingLanguage
\Data {Script}
\Channel Channel
```

The ScriptType tag will define the interpreter that will be required by the application that receives the following script. In the simplest case an application might interpret keystrokes that could be played to the application. At the other end of the scale, the application might implement a Java interpreter. The protocol may need extra messages to allow the linkservice to query the application to discover the scripting language(s) that it can handle. In its current form the linkservice must either know what languages can be handled, or must accept that the script may be ignored.

5.6.9. HeresNewChannel

This message is sent by the linkservice to an application when it has requested a new connection to the linkservice (see CreateNode below).

```
\Subject HeresNewChannel
\SendDocument True / False
\DocumentNickName [DocNickName]
\Channel Channel
```

The SendDocument Tag is used by the linkservice to request that the viewer follows up by actually sending the contents of the node, as is required by many of the hyperbase variety of linkservices. If the linkservice already knows of the document, and has a nickname for the document, it will return this.

5.6.10. CloseNode

This message is sent by the link service to a node that it wishes to automatically close.

```
\Subject CloseNode
\UpdateNode True / False
\Channel Channel
```

The application is then responsible for closing itself, ensuring that it has updated any anchors and its contents if required by the UpdateNode tag.

5.7. Messages that the Editor/Viewer May Send

5.7.1. GetNode

This message allows the viewer to ask the linkservice to send it the contents of a node. This is only required where the linkservice actually stores the node data, and would typically be sent in response to a LaunchDocument message which had the DataCallBack flag set. The linkservice will respond with a HeresDocument message.

```
\Subject GetNode
\DocumentName DocName
\Channel Channel
```

The DocumentName is redundant when calling back for a node, as the linkservice already knows this. However, this message might also be sent to request a new document.

5.7.2. GetServices

When a viewer has loaded up, it will be necessary for it to ask the linkservice to itemise the set of services it may provide on a menu.

```
\Subject GetServices
\Channel Channel
```

The linkservice should respond with a HeresServices message.

5.7.3. GetAnchorTable

The viewer, while running is expected to maintain the anchor table. It is expected to obtain the anchors from the linkservice by sending the message:

```
\Subject GetAnchorTable
\Channel Channel
```

and the linkservice is expected to respond with a HeresAnchorTable message.

5.7.4. RequestService

Most of the hypertext functionality will be obtained by either clicking on buttons or sensitive areas (which are representations of anchors) or making a selection and requesting an action from the menu, or simply choosing an action from the menu.

In the case where a button is clicked, the viewer will know the service that is required from the anchor table. In the case where an action is chosen from the menu, this will define the required service. Messages requesting services will take the form:

```
\Subject RequestService
\Service Service
\AnchorID [AnchorID]
\LocSpec [LocSpec]
\Presentation [colour,style,display]
\Channel Channel
```

The anchor ID will be empty in many of these messages, for example, when requesting that an anchor is

created on a given LocSpec.

The LocSpec may be empty, for example in the case where a simple action has been chosen from a menu.

5.7.5. Update Anchors

When a document has been edited the viewer is responsible for ensuring that the LocSpecs in the anchor table are changed. Consequently it will be necessary to communicate these changes to the link service.

```
\Subject UpdateAnchors
\Data {\AnchorID AnchorID
\LocSpec LocSpec
\Direction Start/End/Bidirect
\Presentation colour,style,display
\Service Service}+
\Channel Channel
```

5.7.6. CreateNode

A user may wish to register a new document that they are currently viewing with the link service. In this case there will not yet be an established channel between the document and the linkserver. The viewer/editor will send the following message.

```
\Subject CreateNode
\DocumentName DocName
\DocumentType MimeType
```

When the linkservice gets the message, it will send a HeresNewChannel message back (see above), and if necessary (SendDocument = True) the viewer will then arrange to send the actual contents of the document using the UpdateNode message. Note that it is quite possible that the document concerned may already be registered with the link service: in this case this message serves only to open a communication channel to the linkserver. Also note that the message contains no information about document ownership or permissions: it is assumed that the linkservice, on receiving this message, will enter into a dialogue with the user to obtain such information.

5.7.7. UpdateNode

If the contents of a node have changed during a session (or if it is a new node) it may be necessary to send the contents back to the linkservice, if the linkservice is of the type that stores the actual contents.

```
\Subject UpdateNode
\DocumentType MimeType
\Data Mime Encoded Data
\Channel Channel
```

5.7.8. Closing

This message is sent when a node is closed by the user.

```
\Subject Closing
```

```
\Channel Channel
```

This will close the channel which may now be re-assigned.

5.8. Messages that both may send

5.8.1. Other

We are very much aware that at times such protocols are insufficient for the needs of a particular problem. We therefore suggest that for experimental purposes, the following extensions to the protocol are allowed. Any message may include a tag

```
\Other Data
```

where the data may be anything that the application and the link service wish to include.

Furthermore, either the application or the linkservice may send a message

```
\Subject Other
\UserDefinedTag User Defined Data +
\Channel Channel
```

so that they can create their own messages with their own data.

Application developers are strongly encouraged to attempt to make the interpretation of these tags and messages optional at the client end, or the applications that they hypertext enable will not be transferable to other systems.

5.8.2. Error

There may be times when a message is sent by one component that the other component does not understand or is for some reason unable to service. In these cases it may be important that the sending component is informed of the error. Either the linkservice or the editor should send an error message when it receives a message that it is unable to process.

```
\Subject Error
\ErrorSubject Subject tag of error message
\ErrorMessage Message
\Channel Channel
```

The Error Message might be used to display in a message box to the user.

5.9. Protocol Summary

Now that we have introduced the entire protocol, it may be convenient to classify the types of messages. Appendix A classifies all messages using this scheme:

- *Linkservice Requests:* messages that the linkserver sends to the client asking for some action.

- *Linkservice Replies:* messages that the linkserver sends in response to all client requests.
- *Client Initialisation Requests:* Messages that the client sends, typically at start-up, in order to initialise itself with the necessary linkservice information.
- *Client Update Requests:* Messages that the client sends to the linkserver to inform the linkserver of changes to the linkservice information that have occurred at the client end.
- *Client Event Requests:* Messages that are sent when a sensitive area is clicked on, or a choice has been made from a link service supplied menu.
- *Client Replies:* Messages that are sent by the client in response to requests from the linkservice.
- *Bi-directional Messages:* Messages that may be sent either way.

6. Example Scenarios

In this section we consider, by reference to some different types of hypermedia viewer / editors, the way that these applications might use the OHP Protocol.

6.1. A Microcosm Semi-Aware Viewer

A Microcosm semi-aware viewer might be a program such as Word for Windows, which has been adapted, usually by the use of application specific macros, to allow the user to make selections and request actions. For the purpose of this example we will assume that we are not going to attempt to enable the application to handle persistent selections: in Microcosm terms this viewer will not handle buttons or specific links, but will allow the user to create local and generic links, and will allow the user to follow such links and to request other services such as computed links.

Let us assume that in this case the user has been working in Word for Windows "off-line", and the scenario begins at the moment that the user decides that they wish to start using hypertext services.

The user selects a menu option on Word asking to register with the link service. This menu might cause the following message to be sent to the link service:

```
\Subject CreateNode \DocumentName C:\\DOCS\\MYDOC.DOC \DocumentType Word4Windows
```

The Linkservice would check to see if it already knew about this document. If not, it would create a new record in the link database registering the document. Assuming that there were already 8 live connections to the link service, it might return the message:

```
\Subject HeresNewChannel \DocumentNickName Hugh's Comments On The
Protocol\SendDocument False \Channel #9
```

The viewer might now request the services

```
\Subject GetServices \Channel #9
```

The linkservice would decide, which would now know the document type, and thus the viewer, and would return the appropriate available services, e.g.

```
\Subject HeresSubjects
```

```
\Data {\Menuitem Follow Link\Service FOLLOW.LINK}
\Data {\Menuitem Show Links\Service SHOW.LINKS}
\Data {\Menuitem Compute Links\Service COMPUTE.LINKS}
\Data {\Menuitem Make Start Anchor\Service START.LINK}
\Data {\Menuitem Make End Anchor\Service END.LINK}
\Channel #9
```

The viewer on receiving this message would build a menu for the user to select these services. Since this viewer is not intended to be capable of handling persistent selections, no request will be sent to the link service for the anchor table.

Now, a user might select the string "making a menu" within the document and select "Make Start Anchor" from the menu. The viewer will send the message:

```
\Subject RequestService \Service START.LINK \AnchorID \LocSpec {\ContentType
ASCII\Content making a menu\Count \ReverseCount \Name \Script } \Presentation
\Channel #9
```

The link service, on receiving this message would note that no LocSpec information, other than the content, had been provided, and would thus start up the link making dialogue with options to create local or generic links. Note that the link making dialogue is within the province of the link service, and is not, currently, part of the OHP protocol.

The user might continue to request further services in a similar way to the above, and then when finished would close the document, which action would cause the following message to be sent.

```
\Subject Closing \Channel #9
```

6.2. Microcosm fully aware

Now let us consider the case of Word for Windows adapted as a fully aware Microcosm viewer. In this example perhaps the user follows a link to a node that is a Word for Windows document held on the network file system. Thus the linkservice might begin by sending the message:

```
\Subject LaunchDocument \DocumentName N:\\docs\\file.doc \ReadOnly False
\DocumentNickName \DocumentType Word4Windows \DataCallBack False \Channel #10
```

indicating that the linkservice wishes Word for Windows to be launched with the named document in editable mode. The linkservice has allocated channel 10.

When word has been launched it will send back the messages:

```
\Subject GetServices \Channel #10
\Subject GetAnchorTable \Channel #10
```

The linkservice might reply to this by sending the same HeresServices message as in the previous section, and also send the message:

```
\Subject HeresAnchorTable
\Data {\AnchorID id_1 \LocSpec {\ContentType ASCII \Content bank balance \Count 1045
```

```
\ReverseCount 2345 \Name \Script } \Direction Start \Presentation 1,1,1 \Service
FOLLOW.LINK}
\Data {\AnchorID id_2 \LocSpec {\ContentType ASCII \Content html editors \Count 2085
\ReverseCount 1305 \Name \Script } \Direction Start \Presentation 1,1,1 \Service
COMPUTE.LINKS}
\Data {\AnchorID id_3 \LocSpec {\ContentType ASCII \Content Dr Smith \Count 3045
\ReverseCount 345 \Name \Script } \Direction End \Presentation 1,1,2 \Service Nil}
\Channel #10
```

In this example three link anchors are described. The first two are start anchors, and the third is a destination anchor (which has a service of Nil). The LocSpecs of these anchors describe the persistent selections in terms of byte offsets through the file, in both directions. An interesting problem arises here. Word can handle and save bookmarks, which would make an ideal method for representing persistent selections. However, if the user who is making links within the document does not have write permission for this document, then it will not be possible to save the document with the bookmarks embedded. For this reason the viewer is not attempting to save the bookmarks with the file, but is saving their position within the anchor's LocSpec, and might turn these positions into bookmarks during the period that the file is being viewed or edited.

The first thing that the viewer would need to do would be to put the bookmarks into the given positions. It will need to check that the content at these position does match the expected content, and if not it will need to apply some algorithm to find where the bookmarks should be placed.

The linkservice might follow it original LaunchDocument request with:

```
\Subject DisplayLocSpec {\ContentType ASCII \Content Dr Smith \Count 3045
\ReverseCount 345 \Name \Script }\Presentation 1,1,1\Channel #10
```

The viewer will now need to highlight the string "Dr Smith" and scroll the document so that this destination anchor is within view.

Maybe the user now clicks on the bookmark over the string "html editors". In this case the viewer will send the message:

```
\Subject RequestService \Service COMPUTE.LINKS \AnchorID id_2\LocSpec\LocSpec
{\ContentType ASCII \Content html editors \Count 2085 \ReverseCount 1305 \Name
\Script }\Presentation \Channel #10
```

This will cause the linkservice to apply the COMPUTE.LINKS service to the given LocSpec.

Before the viewer is closed, it will be the responsibility of the viewer to send an UpdateAnchors message to the linkservice to ensure that all its LocSpecs coincide with the current position of the bookmarks.

Before leaving Microcosm, it is worth making two particular points.

1. Microcosm does not store anchorID's. A Microcosm link consists principally of the start and end LocSpecs. The anchor id's in the anchor table were inserted by the protocol shim for the sake of complying with the OHP protocol. Microcosm itself will never use them: it will communicate with the shim entirely in terms of LocSpecs, and will never send the DisplayAnchor message.

2. It is important that the viewer does not make any assumption that the set of bookmarks it has at any particular time represents the entire set of available bookmarks. This is because Microcosm allows the installation of different sets of links (or Webs), and there may be other anchors within the document that are not currently available. This feature will be common to any distributed link service.

6.3. Classic Dexter Systems

Many of the classic Dexter systems expect that the application will own the anchor table. Such systems might store this table with the data, or might arrange to store the table outside the data in a separate but associated file. When a new anchor is created, its identifier is allocated by the viewer application and is unique within that context. The links that are stored within the linkservice have ends which consist of a (Document Name, AnchorID) pair. This situation can easily be handled by the OHP protocol as follows.

The OHP protocol shim should be written to handle the storage and delivery of the anchor table for each document. Thus when the GetAnchorTable message is received by the protocol shim, it will retrieve the correct file (which it will identify from the channel information). The linkservice itself will never see this message. The viewer will communicate to the linkservice about anchors in terms of the anchorID and the channel (and perhaps the LocSpec), and the shim will convert this into messages about the anchorID and the document name, as shown in figure 3.

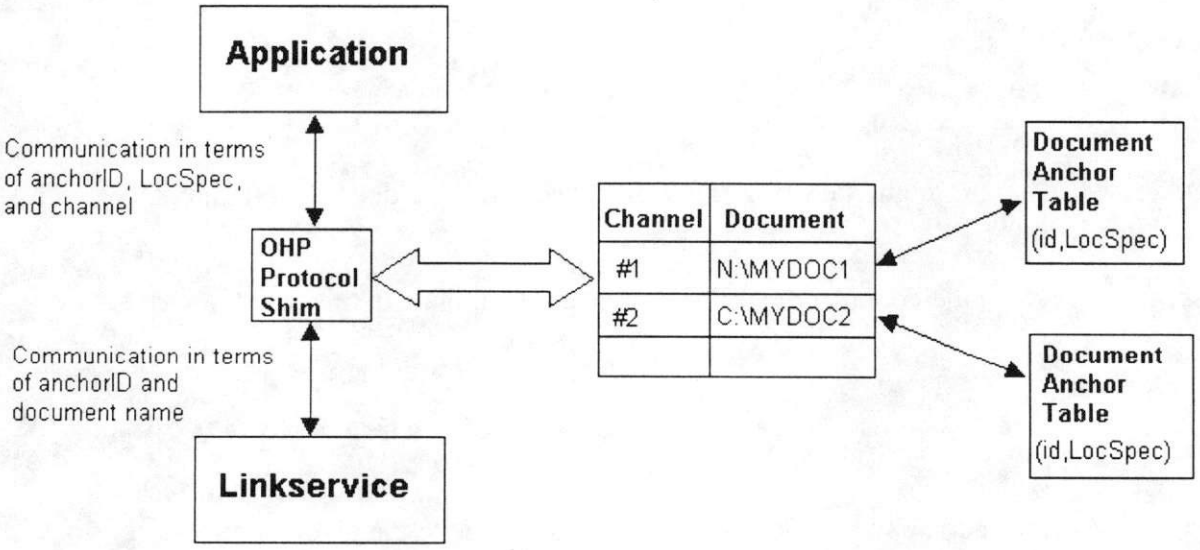


Figure 3: Anchor Table Handling in Classic Dexter Systems.

Since such linkservices do not normally provide a service for allocating anchor identifiers, the protocol shim will need to provide the service instead. When the viewer sends a service request for a new anchor, the OHP shim will consult the appropriate anchor table and allocate the next available anchorID, which it will pass back to the viewer.

6.4. Hyperbase Systems

Most hyperbase systems differ from the descriptions given so far in two respects:

1. They expect to allocate anchors that are globally unique, and they provide a service to allocate such identifiers. The linkservice must of course still know which document an anchor belongs within so that it can display the correct document when a link is followed to the anchor.
2. They often expect to store the node contents. The protocol has the HeresDocument and Update node messages specially to deal with this case. In many cases the node contents may contain the (anchorID, LocSpec) table. If this is the case, the OHP protocol shim will need to arrange to separate the table from the raw node contents, so that the application may first load the contents, then call for the anchor table which will be sent on in a separate HeresAnchorTable message.

6.5. Embedded Anchor Viewers

Where a viewer is being specially built for a format that will only be used hypertext systems it is likely that the developer will specify data format in such a way that the persistent selections may be held as mark-up within the data. Such a format is htf used by Hyper-G. In such cases the anchorID will be encoded into the mark-up and passed to and from the link service to identify the ends of links. These anchorID's might be unique to the document or to the link service as a whole. It is important to realise, however, that not all the anchors held in the document will necessarily be currently valid. In many systems separate Webs of links may be installed, and the anchors that will be valid will depend on the choice of Web.

Therefore, if such a viewer is to work with the protocol it must still send the GetAnchorTable to the linkservice: only those anchors which the linkservice returns should be offered to the user.

7. Existing Standards

A question that has often been posed to the open hypermedia research community is, why do we not use existing standards such as OLE and CORBA? The usual answer has been that these standards do not provide the correct services, and anyway they are by no means standard across architectures, or are not in common use. It is worthwhile, however to consider at this point how existing and emerging standards might be of use in implementing an application to work with a protocol such as OHP.

The prevailing problems, when it comes to hypertext enabling an application are:

- how do we change the behaviour of the program so that it will send and receive the necessary messages? We can usually only achieve this is if the program has an internal macro language or else we can get access to the source code to extend its behaviour.
- how do we get the required information for the LocSpec? This requires that the application allows us to understand something about its data format.
- how do we indicate the presence of anchors within the application data? This requires that we are able to alter the display characteristics of the application.

Existing standards which are well worth noting are the Apple Event Object Model (AEOM) which is a Macintosh standard, and OpenDoc which is a multi-platform model which will shortly be able to interoperate with OLE.

7.1. The Apple Event Object Model

The Apple Event Object Model (AEOM) consists of a number of standardised sets (or suites) of messages and abstract data objects, which are intended to cover both the inter-application exchange and manipulation of most forms of data, and the external control of applications.

The Required Suite is a set of four basic messages which every Macintosh application is required to support. These are: Open Application, Open Document, Print Document and Quit Application.

The Core Suite must be supported by every AEOM compliant application. The messages allow chiefly for the retrieval and modification of data within applications, plus closing documents, and the making of selections.

Other suites are defined for specific data types, including text, pictures, tables, Quicklime and sound.

Most suites define a set of objects to cover the data type that they handle. An application's data is arranged as a hierarchy of these objects. For example: at the top of the hierarchy is the "application" object. This will have a set of "window" elements. A window within an application can be specified by name, or by order: window 1 will be the frontmost, window 2 the second from the front, etc. A window has a number of properties, including name, position, bounding rectangle etc., and selection. It is therefore very easy to retrieve the user selection from the front window of any compliant application. Furthermore, the calling application can specify the form in which it wants the data: it can ask for the content of the selection, or for its offset and range, or for any other attributes that are applicable. If an application is unable to produce the data in the necessary type, it may substitute an alternative, but the form of data supplied is always specified in its reply.

The AEOM is in many ways ideally suited for use with OHP. LocSpec requirements can be translated almost verbatim into AEOM queries, which can be sent to a target application. The application need not be written with any knowledge of OHP; it must simply be written to take full advantage of its own target platform. The AEOM is open to the creation of new suites, so an OHP suite can be defined and used to create applications with a higher level of awareness, using only established tools and techniques.

7.2. OpenDoc

OpenDoc is a multi-platform component-based architecture, based around CORBA. OpenDoc uses compound documents, where the data in each component is viewed and accessed by means of "part editors". OpenDoc assigns the correct editor to handle each different data type, looking first at user preferences, and failing that, making an intelligent decision based on the data type, and its knowledge of the available editors. The part editors share windows, menus and other resources within a document. OpenDoc provides a highly versatile persistent storage system known as Bento, which allows multiple part editors to share a single document file. The Bento storage units and mechanisms are also used for clipboard transfers, drag-and-drop, and linking between parts, documents and/or networks.

OpenDoc includes the *Open Scripting Architecture* - a highly versatile mechanism that includes an AEOM style communication architecture. Consequently the same kind of Shims that would work with the AEOM would work with properly built OpenDoc part editors, and hence with OpenDoc applications. OpenDoc is a multi-platform architecture, and if it is successful in establishing itself the

implications are significant.

8. Conclusions

This paper has described the first draft of level 0 and level 1 of a protocol for client viewers to communicate with a link service. No doubt, as developers try to match the protocol to their own link service, deficiencies will emerge and enhancements will be required.

It is intended that almost any application may run as a level 0 viewer (launch and run with no hypertext), and that level one, as described here, will be the sort of functionality that should be possible to achieve by adapting third party applications. We envisage that level 2 of the protocol will be a much heavier weight implementation.

There are a number of ways in which we can imagine the protocol expanding in the future.

1. The system described here is a fairly passive form of linkservice: clients send requests and get answers which they display. However, systems such as Multicard (Rizk & Sauter) which run server end scripts tend to be much more active, interacting with and changing the data in a node, updating the presentation characteristics, altering the menus interactively, etc. Such behaviour puts a greater onus on the application as there are a much larger number of link service requests that it must be able to handle. We envisage that such extensions will be in level 2 of the protocol.
2. At present the protocol allows the passing of scripts, both to identify a persistent selection, and also as a message in its own right to carry out some process at the client end. At present the language of the script is not prescribed, and the linkservice must know what sort of script to dispatch to a particular viewer. It is possible that these scripts could be standardised in level 2 of the protocol.
3. There are a number of other cases where the link service needs to communicate with the user. For example, as link anchors are identified it is necessary for the linkservice to send the user a dialogue for eliciting such information as the link attributes. Also, when a document is registered with the system the linkservice will need to be given the document attributes. Such dialogues will be highly dependant on the particular link service, but it is quite possible that the protocol could be extended to include the communication of these dialogues to some standard demon at the client end.
4. The set of services which a link service can respond to is not defined by the protocol. It might be possible to identify a set of services that is a superset of all known systems. It is not clear, however that there is much benefit in this exercise.

The best known hypertext system is without doubt the World Wide Web, and the standard html viewers played an important part in making the Web so popular. The Web does not currently use a link service, but there is plenty of research aimed at producing link services for the Web, e.g. the Distributed Link Service (Carr et al, 1995), and once the links are abstracted from the documents, then the Web will no longer be dependant on html to provide hypertext functionality. It will become possible to use any viewer. Perhaps this protocol could have a future in providing a standard interface to Web link services.

Appendix: Summary of Protocol

LocSpec Definition

```
{\ContentType \Content \Count [] \ReverseCount []\Name [] \Script []}
```

Presentation Specification

```
\Presentation
```

where the data that follows the presentation tag is opaque, and will be understood only by the viewer that created it in the first place.

Messages Classified by Type

Linkservice Requests

(messages that the linkserver sends to the client asking for some action.)

```
\Subject LaunchDocument \DocumentName \ReadOnly \DocumentNickName [] \DocumentType
\DataCallBack \Channel

\Subject DisplayAnchor \AnchorID \Presentation \Channel

\Subject DisplayLocSpec \LocSpec \Presentation \Channel

\Subject Interpret \ScriptType \Data {} \Channel

\Subject CloseNode \UpdateNode \Channel
```

Linkservice Replies

(messages that the linkserver sends in response to all client requests.)

```
\Subject HeresServices \Data {\Menuitem\Service }+ \Channel

\Subject HeresDocument \Data \Channel

\Subject HeresAnchorTable \Data {\AnchorID \LocSpec \Direction \Presentation
\Service}+ \Channel

\Subject HeresNewAnchor \Data {\AnchorID \LocSpec \Direction \Presentation \Service}
\Channel

\Subject HeresNewChannel \SendDocument \DocumentNickName [] \Channel
```

Client Initialisation Requests

(Messages that the client sends, typically at start-up, in order to initialise itself with the necessary

linkservice information.)

```
\Subject GetNode \DocumentName \Channel

\Subject GetServices \Channel

\Subject GetAnchorTable \Channel

\Subject CreateNode \DocumentName \DocumentType \Channel
```

Client Update Requests

(Messages that the client sends to the linkserver to inform the linkserver of changes to the linkservice information that have occurred at the client end.)

```
\Subject UpdateAnchors \Data {\AnchorID \LocSpec \Direction \Presentation \Service
}+ \Channel

\Subject UpdateNode \DocumentType \Data \Channel

\Subject Closing \Channel
```

Client Event Requests

(Messages that are sent when a sensitive area is clicked on, or a choice has been made from a link service supplied menu.)

```
\Subject RequestService \Service \AnchorID [] \LocSpec [] \Presentation \Channel
```

Client Replies

(Messages that are sent by the client in response to requests from the linkservice.)

There are none in Level 1 of the Protocol

Bi-directional Messages

(Messages that may be sent either way.)

```
\Subject Error \ErrorSubject \ErrorMessage \Channel

\Subject Other \UserDefinedTag +\Channel Channel
```

There may also be an optional extra tag on the end of any message.

```
\Other
```

Levels of the Protocol

Level 0: Only Supports LaunchDocument and CloseNode messages from linkserver to client. No communication. No hypertext. Any application should be able to run as a Level 0 viewer.

Level 1: Supports the functionality described in this paper. It is intended that it will be possible to achieve this level by adapting third party applications.

Level 2: Not yet defined, but envisaged to support extensions to deal with content manipulation, presentation and standard client end scripts. It is expected that this level of conformity will only be achieved by writing specialised applications.

Responsibilities of the Shims

Server Side Protocol Shim.

(Produced by Link Service suppliers).

- Accept Messages from Link Server, convert to OHP equivalent and pass to correct Client.
- Accept Messages from Client side in OHP, convert and pass to link service in private protocol.

Client Side Communication Shim.

(One needs to be produced for each client platform, for each communication protocol conversion)

- Accept OHP messages from network and communicate message to correct viewer/editor.
- Accept Messages from viewer editor and forward to linkservice in network protocol.
- Maintain table of applications on this platform which should be used for each data type.
- Maintain table showing what level of protocol can be expected from each viewer/editor.
- Handle LaunchDocument messages at the client end, and, if a DataCallBack is required, send the GetNode message and handle the transfer of the data from the HeresDocument message to the application.
- For level 0 applications, handle CloseNode messages, and return error messages for other linkservice requests.

References

Andrews K., Kappe, F. & Maurer, H. *Hyper-G: Towards the Next Generation of Network Information Technology*. Journal of Universal Computer Science, April 1995.

Aßfalg, R. (ed.). *The Proceedings of the Workshop on Open Hypertext Systems, Konstanz*, May 1994.

Carr, L.A., De Roure, D., Hall, W. & Hill, G.J. *The Distributed Link Service: A Tool for Publishers, Authors and Readers*. In: The Fourth International World Wide Web Conference Proceedings. pp 647-656. O'Reilly & Associates, Dec 1995.

Davis, H.C., Hall, W., Heath, I., Hill, G. & Wilkins, R. *Towards an Integrated Information Environment with Open Hypermedia Systems. In:* D. Lucarella, J. Nanard, M. Nanard, P. Paolini. *eds. The*

Proceedings of the ACM Conference on Hypertext, ECHT '92 Milano, pp 181-190. ACM Press, 1992.

Davis, H.C., Knight, S.K. & Hall, W. *Light Hypermedia Link Services: A Study in Third Party Application Integration.* In: The ACM Conference on Hypermedia Technology, ECHT '94 Proceedings. pp 41-50. ACM. Sept. 1994.

Davis, H.C. *To Embed or Not to Embed...*, Communications of the ACM, Vol 38(8), pp 108-109. August 1995.

DeRose, S.J & Durand, D.G. *Making Hypermedia Work: A User's Guide to HyTime.* Kluwer Academic Press. 1994

Grønbæk, K. & Trigg, R.H. *Design Issues for a Dexter-Based Hypermedia System. In:* D. Lucarella, J. Nanard, M. Nanard, P. Paolini. *eds. The Proceedings of the ACM Conference on Hypertext, ECHT '92 Milano*, pp 191-200. ACM Press. Nov. 1992

Grønbæk, K. & Trigg, R.H. *From Embedded References to Link Objects: Toward a New Data Model for Open Hypermedia Systems.* To be published in The Proceedings of Hypertext '96. ACM 1996

Halasz, F. & Mayer, S. (edited by Grønbæk, K. & Trigg, R.H). *The Dexter Hypertext Reference Model.* Communications of the ACM. pp 30-39. 37(2). Feb. 1994.

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

Kacmar, C.J. & Leggett, J.J. *PROXHY: A Process-Oriented Extensible Hypertext Architecture.* ACM Trans. on Information Systems, 9(4) pp 299-419. Oct. 1991.

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

Østerbye, K. & Wiil, U.K. *The Flag Taxonomy of Open Hypermedia Systems*, To be published in The Proceedings of Hypertext '96. ACM 1996

Rizk, A. The M2000 Protocol Description Manual. Euroclid. (Available from the author) 1991

Rizk, A. & Sauter, L. *Multicard: An Open Hypermedia System.* In: D. Lucarella, J. Nanard, M. Nanard, P. Paolini. eds. *The Proceedings of the ACM Conference on Hypertext, ECHT '92 Milano*, Italy, December 1992, pp 181-190. ACM Press. 1992

Wiil, U.K & Østerbye, K. (eds.). *The Proceedings of the ECHT '94 Workshop on Open Hypermedia Systems, Edinburgh, Sept. 1994.* Technical Report R-94-2038. Aalborg University. Oct. 1994.

Wiil, U.K. & Leggett, J.L. *The HyperDisco Approach to Open Hypermedia Systems*, To be published in The Proceedings of Hypertext '96. ACM 1996.

The Notion of Active Object-Oriented Databases for Open Hypermedia Systems (OHS)

position paper for the 2nd Workshop on OHSs at Hypertext '96

Siegfried REICH
Department of Information Systems
University of Linz
A-4040 Linz, Austria
e-mail: reich@ifs.uni-linz.ac.at

January 15, 1996

Abstract

Open Hypermedia Systems (OHS) are characterized by distributed, heterogeneous pieces of information. To cope with the main challenges of OHS one has to tackle consistency, efficiency, and interoperability issues. In this position paper we propose an OHS architecture based on an active object-oriented database system as well as SGML/HyTime as standardized interchange formats. By using active object-oriented database systems consistency, and efficiency in data representation is ensured. By using international standards interoperability of hypermedia information systems is reached.

1 Issues of Open Hypermedia Systems

Open Hypermedia Systems (OHS) incorporate one of the basic ideas of hypermedia technology, i.e., to interconnect different sources of information. Thus two main issues to be addressed can be identified in the literature [17]; namely reference models and frameworks as basic architectures, and standards for hypermedia exchange.

- Hypertext reference models and frameworks address the issue of open hypermedia systems by providing generic frameworks that ensureing the idea of interoperability. These frameworks allow the integration of different hypermedia systems by flexible, multi-layered architectures. There exist several approaches for refernce models and frameworks of hypermedia systems. For an overview see e.g. [4, 17].
- In comparison to models and frameworks, standards for hypermedia interchange have a somehow different objective in achieving interoperability. Their main goal is to define agreed structures and formats for hypermedia information in order

to allow the interchange between different systems. Additionally, as technology develops – for example, in the move to all-digital media – everyone must keep abreast of rapidly changing production methods and quality standards. However, in spite of the bewildering array of multimedia formats, and the skills demanded to employ them, we consider SGML (Standard Generalized Markup Language [11]) and HyTime (Hypermedia/Time-based Structuring Language [12]) as most essential due to their generic and fundamental nature.

Based on experience with existing refernce models and frameworks as well as hypermedia standards we introduce an architectural framework based on active object-oriented database systems and SGML/HyTime to cope with consistency, efficiency, and interoperability issues.

2 Introducing Active Object-Oriented Databases

So far, hypermedia systems have been mainly built based on relational database technology, e.g. [15]. The more flexible and expressive object-oriented model, however, seems more appropriate for hypermedia [4, 5] due to the intuitive and thus more efficient data representation in terms of objects. In addition, hypermedia systems require scheduling and constraint management. Traditional database systems – including (standard) object-oriented systems – cannot adequately cope with these requirements. The main reason being that a polling strategy is not sufficient because either the polling is too late or it wastes resources due to ongoing polling. A possible solution to these problems are active object-oriented databases.

The knowledge of how to react to certain events (i.e. to be *active*) is represented internally in the database system by ECA-rules, ECA standing for Event-Condition-Action. Thus, an event causes under certain conditions a specified action to be executed. ECA triplets are also called triggers.

Examples of ECA-rules in open hypermedia systems could be

- Deletion of a node and notification of references; e.g., the event 'deletion of node 4711' under the condition 'true' causes the action 'send notification message to all nodes referencing node 4711' to be executed.
- Insertion of a new node; e.g., the event 'insertion of node 4712' under the condition 'notification=true' causes all nodes having their notification attribute set to 'true' to be notified.
- Caching; e.g., a set of objects has been requested 'very often' (according to access statistics). Thus the event '64th access of the set of objects 4713' under the condition 'optimizeCaching=true' causes the moving of this object set to faster storage media.

Conditions can be composed of boolean expressions as well as queries on the database content ('select all nodes with dangling references'). A condition is satisfied if the boolean expression evaluates to true or if the query on the database delivers a non-empty result.

Active object-oriented databases allow to model ECA-rules both, local to every single (multimedia) object and global to a set of cooperating objects.

3 Research Proposal

This section describes a research architecture for open hypermedia systems based on active object-oriented database technology.

SGML [11] - and the more HyTime [12] - clearly are standards for information interchange in open hypermedia systems. Various approaches have been made to make databases - especially object-oriented databases - SGML aware, i.e., to allow them to deal with SGML conformant documents (see e.g. [1, 16]).

We are currently developing a system called TriGS$_{SGML}$. TriGS is an active extension of the commercial object-oriented database system GemStone developed at our department [13]. GemStone is a Smalltalk-80 based object-oriented database system. To be able to intuitively handle SGML documents in TriGS we are extending it with functionality for dealing with structured documents encoded in SGML. Thereby, we make use of OPAL's[1] meta class mechanism. This allows us to generate classes for element types at runtime and thus to add specific type information to each instance of an element type. Figure 1 expresses these different approaches. Rounded elements depict instances, rectangular boxes are classes. The main difference is that by the use of meta class mechanisms we are able to dynamically generate a GemStone class for each corresponding SGML element type during runtime (see left part of figure 1).

Our implementation is based on previous experience made by two approaches we have done by using the C++ based SGML-library HyMinder and by doing an implementation in VisualWorks\Smalltalk [7]. The latter uses a public domain Perl-SGML parser. The main advantage of the latter approach is that for each element type a dedicated class is automatically generated during the parsing process thus resulting in type awareness for each instance of an element type. By that, queries on a certain element type are much more efficiently processed because they can be restricted to one class.

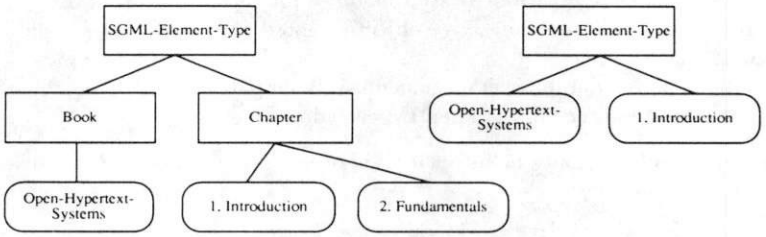


Figure 1: Different Approaches, with and without Meta Class Mechanism

Our research framework allows us to deal with arbitrary types of information because we are able to deal with any SGML document-type-definition (dtd)[2].

In addition to that our implementation forms the basis for a future HyTime engine[3]. In a first step we are currently working on the implementation of parts of the HyTime 'hyperlinks' module which we think is especially important for our

[1] OPAL is the name of the Smalltalk-80 like programming language of GemStone.

[2] By defining a meta dtd which is a dtd defining dtds we are able to treat dtds like 'normal' documents. The meta dtd has been developed by *Hüser* [10].

[3] A 'HyTime engine' is "a program (or portion of a program or a combination of programs)

purposes. The implementation is based on previous experience we have made with the development of HyTime architectural forms in the application area of workflow management [6, 8, 14].

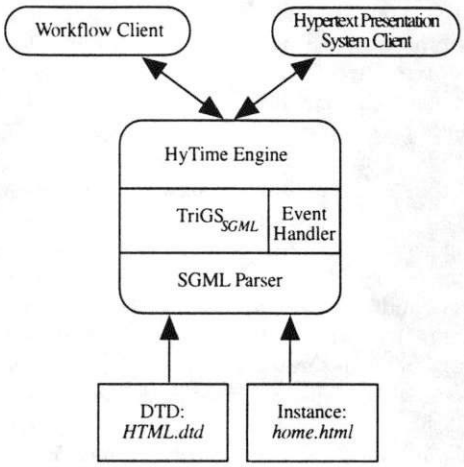


Figure 2: Prototype System Architecture

Figure 2 shows the proposed architecture of our prototype system. On top we have different applications which require link-, constraint-, and scheduling-management such as the already mentioned workflow management system [6, 8] or e.g. a hypermedia presentation system such as the World-Wide Web. These applications are based on a HyTime engine with dedicated modules for each application area. The engine is implemented on top of our active object-oriented database management system TriGS$_{SGML}$.

We consider active database management systems particularly to be able to cope with the following requirements of open hypermedia systems [9, 13]:

- Openness and distribution: integration of other (geographically dispersed) sources of information.
- Support for collaborative work: databases in general support collaboration by providing multi-user facilities. Additionally, active databases can be used for supporting notification and awareness of users.
- Data integrity: by providing storage management and data management facilities as well as management of links/references.
- Dynamism: in order to dynamically reconfigure the object network.

Based on our experience with SGML/HyTime, object-oriented modeling and ECA-rules we can state that the usage of the international standards SGML and HyTime

that recognizes HyTime constructs in documents and performs application-independent processing of them." [12, page 5]

as interchange formats combined with well grounded object-oriented database technology which has been extended by active mechanisms will allow us to better manage structured information in open hypermedia systems.

Within this architectural framework we are currently working on the following most pressing research issues:

- External events: the ECA-rule mechanism of TriGS$_{SGML}$ handles only internal events so far. Especially for open hypermedia systems, external events are crucial. Thus we have to consider the following categories of events [2, 3]: *low-level events* such as mouse, keyboard, insert disk and null events (= no other events to report); *operating-system events*, e.g., if an application is sent in background the operating system sends a background event; and *high-level events* which are events between applications for communication; e.g., requesting a dictionary application for information of a particular word. Thus a protocol is necessary. A special system dependent event-handler will be used for management of external events.
- Transaction management: transaction management for active object-oriented database systems is an open question so far. Nested transactions seem unavoidable, for example, to deal with complex events raised in different transactions. Furthermore, transaction management of the executed 'action' within an ECA-rule – which may not only be a database function but any user- or system-defined function – is subject to research.

References

[1] ABERER, K., BÖHM, K., AND HÜSER, C. The prospects of publishing using advanced database concepts. In *Electronic Publishing, Document Manipulation and Typographie (EP '94)* (Chichester, 1994), John Wiley & Sons Ltd, pp. 469–480.

[2] APPLE COMPUTER. *Inside Macintosh*, vol. VI. Addison-Wesley, 1991, ch. The Apple Event Manager.

[3] APPLE COMPUTER. *Inside Macintosh*, vol. VI. Addison-Wesley, 1991, ch. The Event Manager.

[4] BALASUBRAMANIAN, V. Hypermedia issues and applications: A state-of-the-art review. Tech. rep., Graduate School of Management, Rutgers University, New Jersey, 1993. available as http://www.isg.sfu.ca/~duchier/misc/hypertext_review.

[5] BÖHM, K., ABERER, K., AND HÜSER, C. Introducing D-STREAT. The impact of advanced database technology on SGML document storage. *<TAG> newsletter* (1994).

[6] BURGER, F., QUIRCHMAYR, G., REICH, S., AND TJOA, A. M. Using HyTime for modeling publishing workflows. *ACM SIGOIS Bulletin Special Issue: Business Process Management Systems: Concepts, Methods and Technology 16*, 1 (Aug. 1995), 39–45.

[7] BURGER, F., AND REICH, S. Design and implementation of an abstract SGML interface in Smalltalk. *Computer Standards & Interfaces. Special Issue on SGML* (1995).

[8] BURGER, F., AND REICH, S. HyTime architectural forms for workflow processing. In *2nd International Conference on the Application of HyTime. Vancouver. Available as ftp.techno.com/pub/HyTime/conferences/August_16-17,_1995,_Vancouver_B.C./papers/workflow.ps* (Aug. 1995).

[9] DITTRICH, K., GATZIU, S., AND GEPPERT, A. The active database management system manifesto: A rulebase of ADBMS features. In *Proceedings of the 2nd Workshop on Rules in Databases (RIDS), Athens, Greece* (Berlin/Heidelberg/New York, sep 1995), T. Sellis, Ed., LNCS No. 985, Springer.

[10] HÜSER, C. Update of the report on a prototypical interface for structured documents and its application to the IEN scenario. Tech. Rep. 75/GMD/IPS/DS/I/047/b0, RACE-Programme, TELEPUBLISHING Project (R1075), May 1993.

[11] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), ISO 8879*, 1986.

[12] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information Technology - Hypermedia/Time-based Structuring Language (HyTime), ISO/IEC 10744*, 1992.

[13] KAPPEL, G., RAUSCH-SCHOTT, S., RETSCHITZEGGER, W., AND VIEWEG, S. TriGS - making a passive object-oriented database system active. *Journal of Object-Oriented Programming (JOOP)* 7, 4 (July 1994), 40–51.

[14] REICH, S. Interoperability of workflow information. In *Proceedings of the Doctoral Consortium at CAiSE '95. Jyväskylä, Finland* (1995), pp. 42–43.

[15] SCHUTT, H. A., AND STREITZ, N. A. HyperBase: A hypermedia engine based on a relational data base management system. In *European Conference on Hypertext Technology (ECHT) 1990* (1990).

[16] VITTAL, C., ÖZSU, M. T., SZAFRON, D., AND MEDANI, G. E. The logical design of a multimedia database for a news-on-demand application. Tech. Rep. TR 94-16, University of Alberta. Dept. of Computing Science, 1994.

[17] ZIZI, M. Open systems, information structuring, and navigation. ECHT94 trip report. *SIGLINK Newsletter 3*, 3 (Dec. 1994), 29–30.

Architecture Support for Content-based Open Hypermedia

Li-Cheng Tai
Visual Computing Laboratory,
Dept. of ECE,
University of California, San Diego, La Jolla, CA 92093, USA
atai@ece.ucsd.edu

January 14, 1996

Abstract

Hypermedia allows integration of diverse information sources in the same environment. Current hypermedia systems operate on files and documents, but truly useful hyperlinking needs to be based on information or content semantics. Content-based links has been possible on textual data, but the architectures of current systems are not adequate for multimedia data. This paper proposes a new architecture for content-based open hypermedia. Utilizing a two layer model and a new construct called the Content Registry, this framework supports content-based hypermedia operation on all documents regardless of the media format. First existing works are surveyed and then details of the new architecture are discussed. Finally future work is suggested.

1 Introduction

Hypermedia systems utilize text, graphics, sound, animation and video, but in existing open hypermedia systems, support for data types other than text is limited. To accommodate the increasing use of information in non-textual forms, it is essential for open hypermedia systems to have an architecture which can handle information in a manner independent of the media type, and independent of information containers (files) and locations within containers.

What are the requirements for an ideal open hypermedia system? It should satisfy the following criteria:

1. It should support hyperlinking to elements of document contents. These elements may be words or sentences in a text file, a region in an image, or a moving object in a video.
2. It should support the integration of diverse applications and documents. The document data should not need to be converted to a special format (e.g., HTML) in order to participate in a hypermedia environment. Documents of different origins can be part of the same hypermedia network.
3. It should allow hyperlinking in terms of the semantics of information. In other words, links to information with the same meaning should be supported, independent of the identities of the documents holding the information. Such links are called "content-based links" because they connect meanings or semantics instead of locations or file identities.
4. It should support smart link generation. Users need not create each link manually. The system should be capable of linking related information items automatically when they entered the hypermedia system.

While fully satisfying these requirements is impossible with today's technology, the above criteria can serve as the goal for open hypermedia system (OHS) research and development. Current OHSs, such as Microcosm [1] and SP3 [7] handle textual data well but their support for multimedia data at the content level is either weak or non-existent. This paper proposes a high-level view of an OHS architecture necessary for content-based hyperlinking and interaction for multimedia data.

2 Existing Works

The standard reference model for OHSs is the Dexter Hypertext Reference Model [4]. It presents an abstract view of the essential elements of OHSs, represented by the run-time layer handling presentation, the storage layer managing the hypermedia network data, as well as the within-component layer representing applications and their documents. It provides high-level specifications of anchors and links and addressing mechanisms to separate document manipulation from the management of hypermedia data. This reference model does not specify how content-based hyperlinking should be supported.

Microcosm and SP3 are the best representatives of current OHSs. Microcosm [1, 2] uses a "filter" model to support open hypermedia functionality. Anchor activation and link traversal are implemented as messages passing through a chain of filter processes, each of which can receive messages and take appropriate responses (such as forwarding to the next filter in chain). A filter, called the "link base" manages the storage of anchor and link information. A novel approach in Microcosm is utilizing the data exchange facilities of the host operating system (for example, the Dynamic Data Exchange services and the clipboard in Microsoft Windows and Apple Macintosh) to "tap into" existing documents, thereby enabling non-hypermedia-aware applications to participate in the hypermedia environment. However, full advantage of the services of Microcosm can be utilized only by "aware" document viewers or applications.

Another approach of Microcosm worth mentioning is the use of text retrieval to implement content-based links. By keeping information in the link base about text contents and locations in documents, Microcosm can support "generic links," anchored to any occurrence of a string in all documents, and "compute links," which uses text retrieval techniques to compute the link destinations dynamically, thus supporting true content-based hyperlinking. One issue with generic links is because anchor locations are represented by the offsets in a file, such links can become dangled after documents are edited. The approach of storing anchor locations in the link base also violates the separation of document and hyperlink management because the details of internal structure of the documents are exposed to the link handlers. Clearly, such approaches are not easily extendable to multimedia documents.

The SP3 hypermedia system uses a process-based model. Elements in this system include "participating applications," components (documents), persistent selections (parts of documents attached by anchors), anchors, links, and associations. This system has a very general model in which anchors and links are themselves processes, and link operations corresponding to communications between these processes and participating applications. Applications keep track of the locations of persistent selections in their documents. Therefore in this system a clean boundary is drawn between applications/documents and the "link service." A database provides storage for associations (specifications connecting anchors and links). Applications must be specially designed to utilizes SB3 services. Content-based linking is in the domain of the applications, and the link service provides no assistance.

3 Characteristics of Multimedia Data

Unlike text, raw multimedia data cannot directly participate in hypermedia operations without proper abstractions. The information brought about by multimedia data is task-dependent. Multimedia information systems or Infoscopes [5] will need domain models to recover information from multimedia data. According to the VIMSYS data model [3], multimedia data will be first segmented to obtain image objects, and them domain knowledge can be used to associate these objects with domain objects and events. These "derived features" enable content-based operations like database queries and hypermedia linking.

4 Architecture for Content-based Hypermedia

Generalizing the architecture of Microcosm and SP3 and taking into consideration the characteristics of multimedia data, an open hypermedia architecture is proposed below.

This architecture uses a two-layer hypertext model [6]. The conceptual layer of the system is a Link Engine which manages information about the hypermedia network. However, the Engine does not hold information about the internal details of documents such as in which frames of a video an object appears. Similar to the "participating applications" of the SB3, the manipulation of the document data is the responsibility of

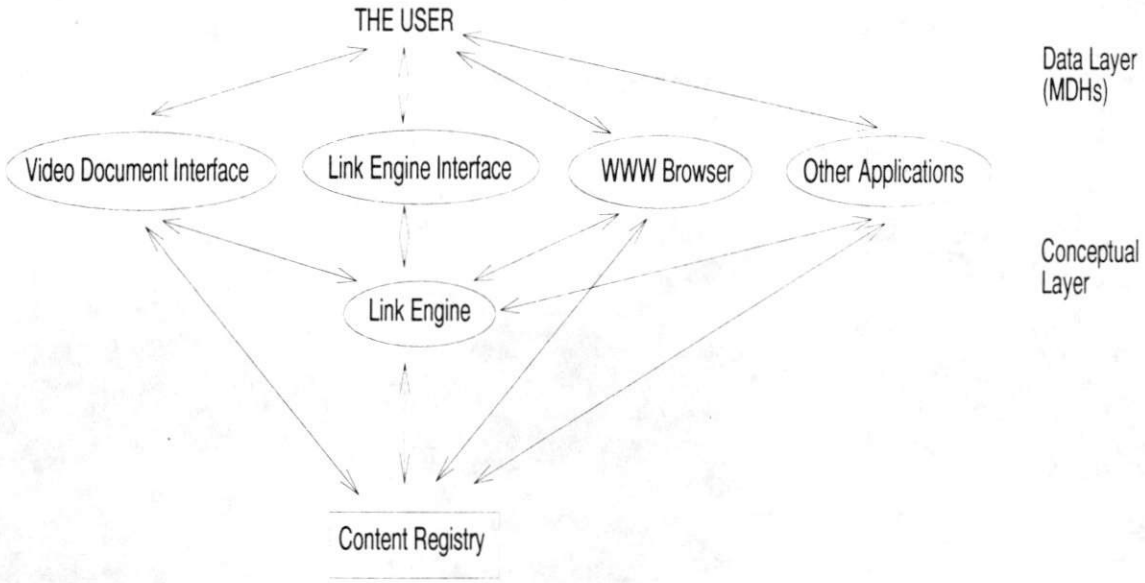


Figure 1: The hypermedia system architecture.

the (data layer) Multimedia Document Handlers, or MDHs. An example MDH is the browser and player of a video database. Other MDH candidates include browsers for the World Wide Web, scientific applications and database front-ends.

A special program, the Link Engine Interface, provides the means for the user to directly interact with the hypermedia network in the Link Engine. It allows link browsing and selection during link traversal and supports visualization and query operations on the network. The Link Engine Interface also handles manual creation of anchors or links by users.

The distinguishing feature of this architecture as compared to other systems is the addition of a central "Content Registry" or CR. All MDHs are required to register with the CR keywords or concept descriptions for elements of the contents in their documents. (The content elements correspond to the "persistent selections" of the SB3.) The descriptions of multimedia data are naturally the derived features from, or metadata associated with, the raw data. The MDHs utilize domain- or task-specific knowledge in description generation. The CR records registry information in a database in the form of

(document name, concept identifier)

Note this tuple does not contain location information like file offsets. It is expected that the MDHs best know how to handle their specific documents and where to find content elements giving a description (concept identifier). The concrete format of the concept identifiers is not specified, but with current technology using textual keywords should be the best choice. Contextual information may have to be included to avoid invoking the wrong document with similar concept descriptions. The Link Engine consults with the CR to find the destination documents during link traversal, as discussed below.

Anchors and links exist inside the Link Engine. An anchor is specified by the tuple (document name, concept identifier). The communication between MDHs and the Link Engine is through anchors. Each anchor maps a data element known to the MDH to a concept identifier known to the CR and thus the Link Engine, as shown in Figure 2. This construct allows a single concept identifier to be linked to many documents (for example, Mary's occurrences in the two video documents are mapped through anchors to the concept Mary). For each anchor the concept identifier and the document name represent the only information required to be known to both the Link Engine and the MDH, and no additional information sharing is mandatory between them.

A link is specified by a tuple (starting concept identifier, starting flag, starting document name, ending concept identifier, ending flag, ending document name). The establishment of a link thus requires the

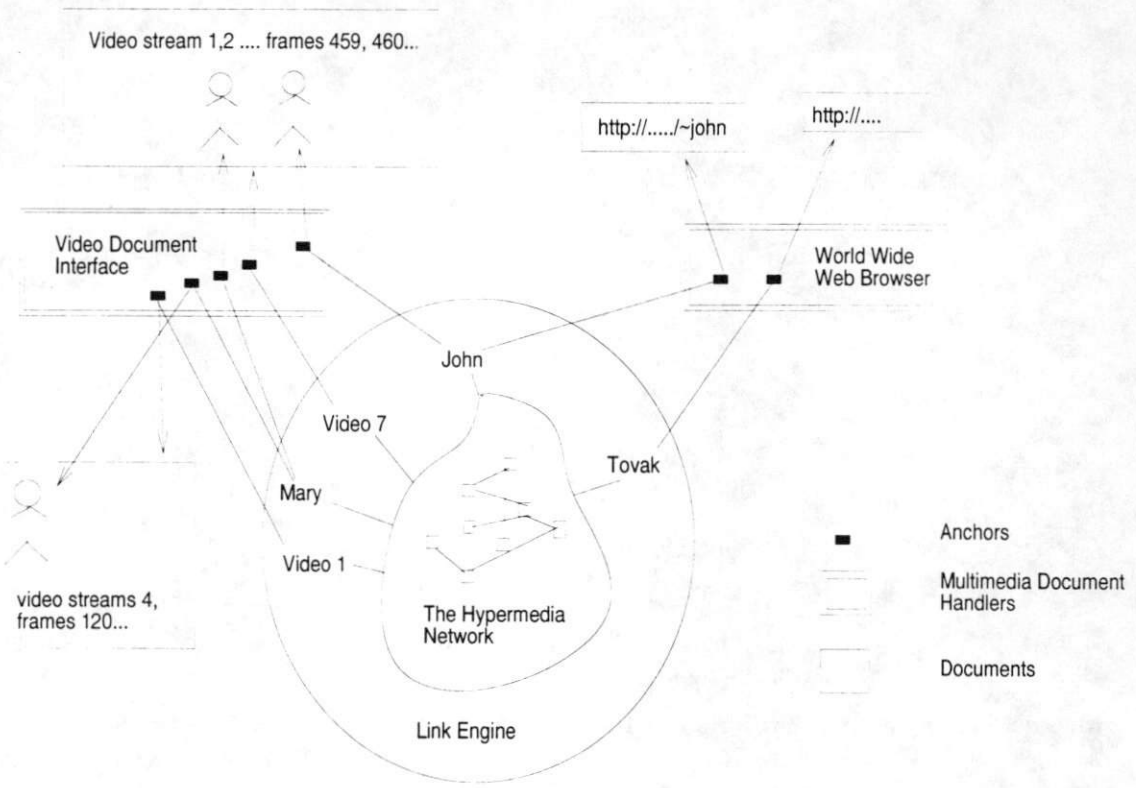


Figure 2: A example scenario showing the relations between MDHs and the Link Engine and the role of anchors.

specification of the starting and ending concept identifiers. The link end points can be declared (in the "flags") to be document-specific (the link is attachable to specific anchors only) or document-natural (the link is attachable to any anchors with the same concept identifier). If a end point is document natural, its "document name" field is meaningless and ignored. Even if the end points are document-specific, using concepts still has the benefit of avoiding dangling links after document modification, as long as the document elements corresponding to the concepts still exist. Although the addressing mechanism of the content elements is left entirely to the MDHs as in the SB3, the traditional type of location-specific links can be modeled by adding location information to the concept identifiers (for example, "Mary in video frame 48") of the end points. These "concepts" will make no difference to the Link Engine but the MDHs can interpret them appropriately. This also means there may be more than one concept identifier for the same content element in a document.

Is the above proposed architecture feasible? With the increasing use of multimedia data, support for content-based hyperlinking becomes necessary. In addition, with the trend toward component-based software and the support of compound documents (OpenDoc and Object Linking and Embedding) incorporated in the operating system (Microsoft Windows and Macintosh), the foundation for a document-based hypermedia architecture is gradually falling into place. Information retrieval technology is fairly mature for textual data, and numerous research works are addressing the development of infoscopes for multimedia data. It only seems reasonable to incorporate intelligent information handling techniques into the document handlers, and to incorporate a central content registry into the operating system to support hypermedia and seamlessly integration of different information sources. The limiting factor for this architecture is the degree of sharing of the various MDHs. The more content is "exported" into the CR, the more useful and powerful content-based operations can become.

5 Conclusion and Future Work

In this paper a high-level view of an architecture for content-based open hypermedia systems is presented. By using a two layer model and the addition of a Content Registry, this architecture provides an uniform mechanism for information in different media formats to participate in a hypermedia environment.

While a general direction is proposed, works are needed to address numerous issues in the realization of this architecture. For example, the language for the concept identifiers need to be developed which can effectively model document contents of different applications. The Link Engine and Content Registry need to organize their data for efficient access so link traversal does not become a bottleneck of the system. The overhead of the CR for keeping track of content changes in documents needs to be minimized. How to scale this architecture up to the network environment is a challenging issue. In addition, automatic link generation is still not addressed.

Despite these open issues, realization of this architecture can have significant benefits allowing intelligent software to operate in information contents seamlessly, currently made impossible by the artificial boundaries of files and data types. For example, the Link Engine can examine its network and do inference on how different information elements are related. The Content Registry can be used to generate abstractions for effective browsing and querying of information. By grouping information from different documents in the same space, new possibilities for content-based information processing are opened.

References

[1] N. D. Beitner, W. Hall, and C. A. Goble. Multimedia support and authoring in Microcosm: an extended model. Technical report, Dept. of Electronics and Computer Science, Southampton, UK, 1994.

[2] H. Davis, W. Hall, I. Heath, G. Hill, et al. Towards an integrated information environment with open hypermedia systems. In *Proceeding of the ACM Conference on Hypertext*, pages 181–190, Milan, Italy, Dec. 1992.

[3] A. Gupta, T. Weymouth, and R. Jain. Semantic queries with pictures: The VIMSYS model. In *Proceedings of the 17th International Conference on Very Large Database*, pages 69–79, Sept. 1991.

[4] F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. *Communication of the ACM*, 37(2):30–39, Feb. 1994.

[5] R. Jain. InfoScopes: Multimedia information systems. In B. Furht, editor, *Multimedia Systems and Techniques*, pages 217–254. Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[6] J. Mayfield. Two-level models of hypertext. In R. D. Sammel, editor, *Advanced in Software Engineering and Knowledge Engineering*. World Publishing, 1995.

[7] J. L. Schnase, J. J. Leggett, D. L. Hicks, P. L. Nurnbery, and J. A. Sánchez. Open architecture for integrated, hypermedia-based information systems. In *Proceedings of the 26th Annual Hawaii International Conference on System Sciences*, volume 3, pages 386–395, Wailea, HI, USA, Jan. 1994.

Open Hypermedia File System

Tom DeDonno 1/8/96
UCSD/ECE-0408
9500 Gilman Drive 220
La Jolla, Ca 92093
(619)433-4170
tdedonno@sdcc3.ucsd.edu

Abstract

This paper outlines the objectives of an open hypermedia file system. A hypermedia file system addresses the objectives associated with the development of a multimedia file system and takes a step further by providing additional capabilities to meet the needs of future hypermedia systems. Standard multimedia file systems have three objectives:

* accommodating large data objects

* meeting continuous interactive playback requirements

* and synchronization of multimedia data

A hypermedia file system should meet both these objectives and addresses other issues arising in hypermedia. More specifically, it should address: the ability to automatically access and index multimedia data, effective handling of dangling links, Quality of Service negotiations over networks, spatial indexing of temporal objects thereby permitting precise random access of temporal data, open hypermedia, creation of generic linking of non textual objects through file annotation, and robust network security providing multimedia object access lists based on network domains.

Multimedia File System

Multimedia file systems are concerned with handling of large contiguous clusters, continuous time constraints and synchronization of heterogeneous media. When dealing with temporal based data (i.e., video & audio) in a multimedia system the file system must send information to the user at a continuous pace. To enable a viewer to watch video at the standard thirty frames per second, the multimedia file system must access and send thirty frames every second to the display. Multimedia file systems have the option of delaying the start of the video. However, once the video has commenced, jitter (variance in time between sequential delivery) must be less than the playback rate of the medium. [2]

Many forms of multimedia data have very high disk space requirements. Table 1 illustrates the disk space requirements for the four MPEG (Moving Picture Expert Group) standards. MPEG stands to become the most popular form of compression for video and audio streams. Traditional file systems hardly ever deal with dynamic data files larger than 1MByte. However, table 1 illustrates that a 90 minute TV quality movie compressed using MPEG-1 has a disk space requirement of 1GByte. The MPEG-3 standard supporting HDTV (High Definition TV) would require 225MBytes of disk space for just a two minute video presentation. Table 2 lists the evolving hardware capacities of server and local PC disks. These tables illustrate that the storage and playback of movie length high quality MPEG videos are unlikely in the near future. A standard 2GByte PC disk in the year 1998, would not fit even a single MPEG-2 hour and half VCR quality movie. However, interpolating from tables 1 and 2, a multimedia system in the year 1998 with a 2Gbyte disk could hold the following multimedia video streams:

* Short HDTV quality MPEG-3 video segments

* Full length MPEG-4 video conferences

* MPEG-1 movies stored predominantly on CD-ROM with hard disk cooperation

Table1: MPEG Storage Requirements

Standard	Quality	Compress Ratios	Duration		Playback	
			2-Minutes	1.5Hour	Block/sec	msec/block
MPEG-1	TV	1.5Mbps	22.5MB	1GB	192	5.21
MPEG-2	VCR	5Mbps	75MB	3GB	640	1.56
MPEG-3	HDTV	15Mbps	225MB	10GB	1920	.52
MPEG-4	Teleconf	25kbps	375KB	16MB	3	333.3

Another key concern of multimedia file systems is the synchronization of related but disparate data. A multimedia file system could create heterogeneous blocks that contain a mixture of all the objects needed to be synchronized. For example, a text stream will have the corresponding audio built into the text based document structure. The system could also store each type of multimedia data in the appropriate format and include additional information in separate files to establish synchronization. The first method is impractical since all forms of multimedia data are being driven in the direction of independent compression formats (i.e., JPEG, MPEG, Text, RTF, etc.). In addition, the second method places the burden of synchronization directly on the file system.

Table2: Evolving Disk Capacities		
Year	PC($250)	Server($2K)
Jan/95	500MB	2GB
1998	2GB	8GB
2001	8GB	16GB

Hypermedia File Systems

The prime objective of a hypermedia file system is to simplify and enhance the performance of a hypermedia system by providing libraries and mechanisms to do the key tasks. A hypermedia system is concerned with combining various forms of media through interlinking yielding an enhanced platform for presentation and dispersion of information. To accomplish this goal a hypermedia file system must inherit and enhance many attributes of the previously discussed multimedia file system. Besides, enhancing a multimedia file systems, a hypermedia file system is devoted to accomplishing two other objectives: Provide a vehicle to aid in the creation of active hypermedia systems. Solve the special needs of future hypermedia systems that will gradually replace the current hypertext (i.e., Mosaic, Netscape and Gopher) systems.

Enhancement of Multimedia File System

As previously, stated multimedia file systems are concerned with: allocation of long adjacent blocks, synchronization of heterogeneous media, and meeting continuous time constraints. To effectively meet the MPEG playback rates of table 1, three basic requirements must be satisfied: specialized graphic hardware, adequate storage transfer rates, minimization of disk access times.

Specialized Graphic Hardware

Future graphic hardware will easily playback MPEG at thirty frames per second. Already many vendors are selling high performance PC graphic cards with thirty frames per second MPEG decoders selling for under $300. For example, the Stealth64 Video 3200 is priced at $299.

Adequate Transfer Rates

The transfer rate of a 1.2GByte IDE disk is 12MBytes/sec (i.e., 12KBlocks/sec) these transfer rate can adequately support all classes of MPEG except the high quality MPEG-3.

Minimization of Disk Access

Unfortunately, disk access times are substantially slower than transfer rates. Permanent storage systems must employ some form of nearby block allocation to guarantee continuous playback. With defragmentation software, computer systems can meet interactive continuous time constraints and still maintain the basic file system structure. A key ingredient for the success of any new idea or system is to maintain compatibility with existing standards and provide enhancements to accommodate the needs of evolving systems. In addition, to maintaining standards, the designer must consider future hardware directions and capabilities. Integrated device electronics is increasing at overwhelming rates. However, mechanical devices are improving very slowly. The main permanent storage device of almost all computers is the hard disk. Hard disks are plagued with two mechanical shortcomings: seek time, disk head movement between concentric tracks; and latency delay, the time it takes for the constant rotating disk platters to rotate to the correct segment of a track. Because of these two mechanical restrictions, disk performance has not kept pace with evolving CPU performance. Disk performance has been doubling every five and half years, with CPU performance doubling almost every eighteen months. Without buffering a system must access, read and transfer each block within the time constraints listed in the last column of table 1. For MPEG-1, this value is 5.21 msec/block. On a modern day disk the time to read, access and transfer one random block at a time is approximately 17.64msec. This value is roughly three times greater than the maximum allowed delay. Nota bene, even if one eliminates the seek time, the corresponding latency delay is still too slow for continuous playback.

Since current disks cannot meet MPEG continuous playback by reading one block at a time will future disk provide the necessary performance? By extrapolating the performance increases between the years 1988 and 1995. A 1999 future disk will read, access and transfer a random block at 9.025msec (i.e., .015+4.4+4.61). Even this future disk would not meet the required MPEG-1 playback rate of 5.21msec. Obviously in order for interactive video to become a reality on future computers the file system must minimize seek times and latency delays. Since disk rotation is purely mechanical latency delays will become the weakest link. The best way to increase performance in anything is to remove the weakest link. In hard disks, the weakest link will eventually become the latency delay. Latency delays can almost be eliminated by positioning all strands (i.e., video segment lasting approximately one second) onto only one cylinder. The blocks would be in ascending order so the entire strand could be read during one rotation. The length of the strand would be sufficiently large enough so that latency delays don't exceed the required playback rate of the device.

Active Hypermedia

Most hypertext systems are passive. An author must manually add links to the system in order for the user to access information. In an active hypermedia system, the linking of information is automatic. A hypermedia file system can aid in the creation of active hypermedia systems by providing five basic features: First, automatic indexing of key hypertext (i.e., HTML) files and homepages. Second, direct communication links with robots, wanders, and spiders. Robots, wanders and spiders are often called web crawlers. These crawlers search computers for key HTML files and send their findings back to a home system. The hypermedia file system can provide key information directly to these crawlers alleviating the requirement that the crawlers physically access each host machine. Third, additional file mappings based on file types not only directory structure. When one is using a MPEG viewer, a file structure showing all nearby MPEG videos is superior to the standard tree structure. Fourth, non textual hypermedia files such as MPEG, JPEG, and GIF cannot be accessed through standard text based queries. Database management systems allow query access to text based system through text based queries. To allow text based query access of non textual data, the hypermedia file system will maintain a corresponding keyword and annotated text file associated with each non textual multimedia object. A query will consist of accessing the annotated files and a selection will result in sending the associated multimedia object. Fifth, current hypertext systems contain only specific links. A specific link is created by an author at a specific location within a specific file pointing to a specific document. Active systems must allow the inclusion of generic and dynamic linking. A generic link allows the author to attach a link to a single instance of an object that is then automatically attached to all instances of the object.[1]

Evolving Hypermedia Systems

Besides satisfying the need of active hypermedia, future hypermedia systems must seek to provide the following capabilities: resolution of dangling links, spatial indexing (i.e., random disk access) of temporal objects, quality of service (QOS) negotiations, enhanced communication with hypertext protocols, and network based security access lists.

Dangling Links

A key problem with standard hypertext systems is dangling links. Almost all HTML files have pointers to other documents. Significant research has already been done in these area. A hypermedia file system must inherit some form of link bases to eliminate dangling pointers.

Indexing of Temporal Objects

Traditional file systems provide random and sequential access of spatial/sequential based data. Skipping ahead one or two pages in a spatial/sequential document is relatively easy on a spatial based medium. However, skipping ahead five minutes into a video is difficult. The video is a temporal based medium and random access to disk are spatial. A system can provide random access to temporal data in one of two methods: First, it can estimate the starting location of a specific frame by considering average compression rates. For example, MPEG-1 compresses at 1.5Mbps, a random access point at 3Mbits would be roughly two seconds into the video. The major drawback of this approach is the wide variance between compressed files. Going from an average compression ratio to a spatial file location is only an estimate. The system could very well deliver video that is nowhere near the desired time frame. In addition, what part of the corresponding audio stream must be sent to maintain synchronization. Second, the hypermedia file system creates an index of a MPEG video similar to a standard IBM VSAM. Each intracoded frame would have an index file entry consisting of a doublet containing file offset and frame number. Both these values are accurately obtained. The physical spatial disk location is a standard file system component. The individual times of each MPEG intracoded frame is obtained by simply keeping track of all frames from the start of the video. MPEG compression consists of intercoded and intracoded frames. An intercoded frame relies on information on previous and/or future intracoded frames to reconstruct itself. An intracoded frame is encoded and decoded with information pertaining to only itself. MPEG viewers can start playback at any MPEG intracoded frame. This method would also lend itself well to fast forward and reverse. In fast forward mode the system can send a specific sequence of Intracoded frames depending on the speed of fast forwarding. Note this method would provide the intrinsic capability of watching an entire video at a fast forward rate even if the corresponding physical hardware was unable to meet this continuous demand. For casual use, the size of the index file can be greatly reduced by generating a ten-second index file. For every three hundred frames HFS records the file offset of the next intracoded frame. Typical viewers would not require a fast access mechanism whose precision exceed ten seconds.

Open Hypermedia

Almost all hypertext and hypermedia systems developed are closed hypermedia systems. A closed hypermedia system requires that the original document be marked up with linking information. In an open hypermedia system such as Microcosm, the link information is maintained in a linkbase. A linkbase is a link database file that maintains information on all links. [1] Future hypermedia systems will need to access and organize entire libraries of information. These libraries of information will prohibit the markup of the originals. The originals may be on read only material or the system would require the author's permission to markup a document. A solution to this problem resides within the hypermedia file system's ability to create independent link bases that perform hyper linking on read only material. When appropriate the hypermedia file system would transparently combine the link bases and read only material.

Quality of Service (QOS) negotiations

A major drawback of the world wide web (WWW) is the slow delivery of images. Users accessing the WWW at non prime hours with a direct digital connection approach interactive access. Users accessing the WWW through the standard analog phone lines are usually burdened with slow image delivery. Unfortunately, most WEB users use slow analog phone connections. Two popular HTML hypertext engines Mosaic and Netscape provide a user based long term scheduler. The Mosaic and/or Netscape screen will display text at an interactive pace, however, images are initially displayed as just an image bar. The actual image is not displayed unless the user physically waits for its arrival. With the coming arrival of video information to the WWW, even users with direct digital connections will wait several minutes for video delivery. Even today, expensive teleconferencing systems are burdened with delays on video delivery. If optical networks (i.e., ATMs) are installed and deliver the promised network bandwidth, Internet delivery of real time video and images may become a reality. However, without optical networks a suitable compromise besides a user based long term scheduler, is to setup a quality of service (QOS) negotiation between the hypermedia file system and the requesting hypermedia system engine. For example, the JPEG standard can compress images at various resolution. On playback the hypermedia engine could request that all images be sent at only 25% resolution. As another example, when the network cannot support video at thirty frames/second, the hypermedia engine and file system can negotiate quality of service of fifteen frames/second.

Network Access Lists

Today computer systems deal with security at predominantly the user level. Computers don't sufficiently handle network security. Network security deals predominantly with allowing certain systems to access only part of the file system. Future hypermedia systems will require a more robust network security system. The Unix operating system associates an access list with each file. An access list consists of a three level user domain (i.e., everyone, group and user) and three file access rights (i.e., read, write and executable). An access list associates a user (general term domain) and access rights to each object. A capability list associates object and access rights to each domain (specific term user). Future hypermedia file systems must extend this concept to a networked environment. The domain associated with a file is extended into a network entity consisting of: network domain (i.e., com, edu), computer network (i.e., UCSD, sun, etc.) and specific network node (i.e., CSE, SDCC3). The access rights would be identical or similar to a standard Unix file system.

Conclusion

Future file systems must position themselves as middle ware applications that lie on top of the operating system kernel. These middle ware applications (e.g., Hypermedia File System) can provide the necessary mechanisms to meet the needs of future application programs and still maintain compatibility with existing computer systems. This middle ware hypermedia file system can meet the needs of multimedia storage and retrieval by providing a cylinder approached method of storing and retrieving video segments. An optimal number of blocks are restricted to exactly one cylinder. These blocks all occur within the same time frame of a video and have a performance limitation of only one latency delay. As these blocks are read off of one cylinder they are sorted and sent to pinned memory buffers associated with the video. Besides multimedia storage and retrieval, these systems must also provide an active hypermedia system foundation, that can automatically integrate new information into existing hypermedia databases. The hypermedia file system can provide active hypermedia foundations by providing: Annotated key files of non textual based objects and generic linking of non textual objects. The final responsibility of a hypermedia file system is to meet the specialized requirements of future comprehensive hypermedia systems. These specialized requirements include: the effective handling of dangling links, spatial based indexing of temporal objects, open hypermedia facilitates, quality of service negotiations, easily accessible network communication, and network based access lists.

References

[1] G.A. Hutchings, A.J.D. Warren & S.T. Rake. Microcosm User Guide Release 3.0 for Windows, University of Southampton, MLS Technology Ltd. May 1994.

[2] Harrick M. Vin and P. Venkat Rangan. Designing a Multi-User HDTV Storage Server. IEEE Journal on Selected Areas in Communications, 11(1):153-164, January 1993.

[3] Jeffrey R. Bach, Santanu Paul, and Ramesh Jain, A Visual Information Management System for the Interactive Retrieval of Faces, IEEE Transactions of Knowledge and Data Engineering Vol. 5, No. 4, Aug. 1993, pp 619-628

[4] Ramesh Jain Editor, NSF Workshop on Visual Information Management Systems, Feb. 24-25 1992, Redwood Ca.

Developing Hypermedia Over an Information Repository

Panos Constantopoulos, Manos Theodorakis and Yannis Tzitzikas
Department of Computer Science,University of Crete
and
Institute of Computer Science,
Foundation for Research and Technology - Hellas
email : {*panos* | *etheodor* | *tzitzik*}@*ics.forth.gr*

Abstract

We propose developing hypermedia applications over an information repository. The main benefits of this approach include development efficiency, product quality, ease of tailoring and extensibility. In particular we focus on design and implementation issues related to presenting, navigating and retrieving large amounts of highly interlinked, complex data. DOMENICUS, a prototype customizable hypermedia engine built on top of a repository system ,the Semantic Index System, is briefly reviewed.

1 Introduction

Improving the quality of hypermedia design and reducing the development cost is an important challenge for the information industry. In addition the integration and collaboration of existing and future tools over a general hypermedia environment would contribute to the efficient usage of the human knowledge stored in computer systems, and would enhance productivity, collaboration, cognition and learning.

In this paper we focus on the design and implementation issues concerning hypermedia systems needed for the presentation of large amounts of highly interrelated, structured, heterogeneous data, for which complex query and retrieval methods are needed and are subject to update (in contrast with permanent data). Actually, these applications are very laborious and this holds for all the steps of their design process and life-cycle: domain analysis, navigational design, interface design, implementation, data entry, testing, adaptation/tailoring, data evolution and system upgrade. Example applications include encyclopedias, scientific catalogs, on-line museum systems and geographical/political atlases.

We use an *information repository* in order to represent the logical structure of the data, as well as information concerning their usage and management (display, retrieval). A repository is defined [1] as a shared database of information about engineered artifacts. We use a repository manager, the Semantic Index System (SIS), which supports rich structuring mechanisms: classification, generalization and attribution, which are indispensable in order to deal with the data/usage complexity. In case of non structured data (images/video/audio/plain text) only their semantic description (if any) and pointers to their storage location are stored in the repository.

On top of SIS we have developed a customizable hypermedia engine, DOMENICUS, which allows presentation, navigation and retrieval of data (including multimedia). It supports a set of core functionalities, needed in most application domains, which are customizable according to a *Presentation Model (PM)*, stored in the repository itself. It supports cards whose contents are determined at run-time (limited natural language generation), are customized easily (by the PM), and offer flexible and consistent hyperlinking. Thus, the development cost (hyperlinking and data entry effort) is minimized and we get fully connected and consistent presentations, which are flexible and extensible, thus, suitable for on-line applications. In addition it offers complex retrieval methods and incremental query formulation. Representing the data in a repository permits efficient exploitation and data reuse.

In section 2 we refer to some crucial issues relating to hypermedia development. In section 3 we describe the Semantic Index System (SIS), our repository manager, and in section 4 we present DOMENICUS. In section 5 we present future directions and in section 6 we draw some conclusions.

2 Crucial issues

Crucial issues concerning hypermedia development are the following:

- **Information Representation**
  Information representation determines the degree of data exploitation. In case of hypermedia applications rich structuring mechanisms are needed to handle data complexity (eg: scientific knowledge and engineering designs or constructs) and usage complexity (complex retrieval methods, usage of the same data by different users for different purposes). Multi-facet classification [8] must be supported.

- **Information Management**
  Advanced information management capabilities are needed in order to result in presentation quality and effectiveness. Data integrity, consistency and privacy should be addressed. Further issues concern collaboration, distribution and interoperability.

- **Extensibility**
  Domain data evolution should be feasible and performed in an easy manner. In addition, data reusability and tool integration and collaboration are indispensable. We agree with [5] that a complete hypermedia environment is needed at operating system level.

Therefore we believe that it would be advantageous for hypermedia development to take place over information repositories integrated with other tools. Centerpiece of a repository is the repository manager. Bernstein [1] defines the repository manager as a database application that supports checkout/checkin (private workspaces), version and configuration management, notification, context management, workflow control and rich structuring mechanisms in order to handle data sharing and interchange.

Moreover integrating repository and tools will release tool developers from implementing tool-specific databases and will enable tools to exploit the amenities of a repository manager.

3 The Semantic Index System

The Semantic Index System (hereafter SIS) [4, 3] is a system for the management of very large collections of highly interrelated information objects with evolving structures. This system is especially well suited for use as a repository system, providing metadata management and the kernel of an integrated environment of a dynamic collection of tools [1].

SIS uses Telos as the information representation framework. Telos [6] is an object-oriented knowledge representation language that supports a number of structuring mechanisms as well as an assertional and temporal reasoning sublanguage. In SIS we confine ourselves to a version of the structural part of the Telos language.

Objects in Telos are named and organized along three dimensions: attribution, classification and generalization. A distinctive feature of Telos , and consequently of the SIS data model, is the uniform treatment of individuals and attributes. This allows attributes to be organized in classification and generalization hierarchies and to have attributes of their own, which provides great expressive power and flexibility.

Multiple classification is allowed, supporting the separate representation of multiple modeling aspects. An open-ended classification hierarchy is possible. Classes within a given instantiation level are also organized in terms of generalization (or isA) relationships. These can be multiple and give rise to hierarchies that are directed acyclic graphs. They induce strict inheritance of attributes, in the sense that inherited attributes cannot be overridden but only restricted by the definition of the subclass.

The SIS outperforms all relational systems on the market in lookup and traversal access times by a factor of about 25. Thus, it currently is a unique pragmatic solution for efficiently handling very large sets of highly structured data.

The SIS user interface supports menu-guided and forms-based query formulation with graphical and textual presentation of the answer sets. It also supports graphical browsing and navigation in a hypertext-like manner. A hypertext annotation mechanism is also provided. Menu titles, menu layout and domain-specific queries are user-configurable. Thus the user interface can be customized to the application without changing the executable code.

A forms-based interactive data entry facility is provided. It allows for entering data and schema information in a uniform manner. By employing the schema information, it automatically adapts itself to the structure of the various classes and subclasses. Furthermore, it is customizable to application-specific tasks, such as classification of items, addition of descriptive elements, etc.

An Application Programming Interface (API) for communication with other tools is provided.

So far, SIS has been used as the kernel for various applications, such as a Software Static Analysis and Class Management System [3], the CLIO Cultural Documentation System [2], and prototype systems for thesaurous management and mechanical fault documentation and diagnosis.

In the framework of the AQUARELLE project [1] SIS (and CLIO) is going to be used for the storage and management of multimedia folders and will be integrated with SGML editors in order to create, store and make accessible documents (folders) with referential integrity to formal knowledge entities and other document parts.

4 DOMENICUS: A repository-based hypermedia engine

DOMENICUS is a hypermedia engine developed over SIS which supports a set of core functionalities appearing in most application domains: alphabetic lists, subject catalogs, guided tours, query cards, hyperlinks, image annotations, bookmarks and history. Since hypermedia applications are addressed to different kinds of users it offers a simple, friendly and uniform interaction with the user.

The repository contains two types of data: (a) *domain data* and (b) *usage data* (used to determine the presentation and management of the domain data). In order to represent the *domain data*, the appropriate semantic network is constructed using the Telos language. In case of multimedia data, only logical pointers to them are stored in the knowledge base. The granularity of structuring should be determined by (a) the queries that should be answerable, and (b) the presentation requirements.

The *usage data* are used to customize the DOMENICUS functionalities, that is, the main data categories (subject catalog), the available guided tours, the contents and the interpretation of the query cards, the presentation card types and their contents (including their hyperlinking).

A *Presentation Card Specification* (PCS) defines a mapping of the knowledge concerning one object of the repository to a set of multimedia information: images, video, audio and

[1] AQUARELLE is a project of the EC Telematics Programme aiming to the sharing of cultural heritage.

a text in a format which is close to natural language (enriched with hyperlinks), making the contents of cards friendly. Card specifications can specify static and dynamic elements (expressed in the supported query language), and are used at run-time in order to produce the card contents. A PCS can be assigned to an object or a class of objects. Exceptions to the presentation declaration of a class are possible , permitting artistic interventions to the presentation of some members of the class.

Hyperlinking is one of the distinctive features of our approach since our presentation model permits the declaration of hyperlink classes (some of the link classes of the base) and hyperlink connections determined by queries. This means that hyperlinks are produced dynamically at run-time, assuming that the information is structured appropriately in the knowledge base. This minimizes the cost of hyperlinking [2], and results in fully connected and consistent presentations, ease of tailoring, and permit data/schema update and evolution (needed for on-line applications). In addition, the data entry/modification effort is reduced. The methodology of hyperlink construction prevents dangling and erroneous links and makes the presentation stable since the hyperlinks [3] are subject to the integrity control of the repository. Moreover hyperlinks are typed (they belong to classes, metaclasses,...) and this can be exploited in order to adapt/filter the card contents easily (even at run-time).

The domain and usage data relationship is represented in the base itself in a manner which is clear, flexible, consistent and can be used efficiently. The same domain data can be related with more than one set of usage data (that is, presentation descriptions). For example, from a single cultural knowledge base, we can provide on-line presentations dedicated to museum curators, WWW users, or we can produce a stable presentation disposed as a CD-ROM.

DOMENICUS can be used for the development of hypermedia applications whose knowledge evolves over time, offering development efficiency, extensibility and easy customization. DOMENICUS has been used for a prototype electronic presentation of the painting exhibition "From El-Greco to Cezane" held in the National Gallery of Athens in 1992.

5 Future directions: Towards an advanced information repository

We keep working with our repository manager (SIS) and its usage for hypermedia development. Regarding SIS, we focus on *context management* [7] and *tool integration*. In particular, we are working with issues regarding context-based object naming [9] which, among others, contributes to the quality of the limited natural language generation (used in DOMENICUS cards) and data entry facilitation. We also study issues concerning view/context updates [10], in order to address filtering, authority and collaboration issues.

Tool integration is crucial since it will permit tools to share/interchange data without the need of special protocols. This needs special translators to map a tool's data format to a canonical format stored in the repository, but is not enough for tools that access data interactively during their execution. This problem can be solved either by modifying the tools in order to communicate with the repository (this is a long term solution which requires accepted standards), or by implicit methods like implementing a virtual repository interface in order to trap all tool's data accesses, translating them to repository accesses.

Regarding SIS and hypermedia, we are currently working on the mapping between SGML and Telos.

---

[2] Actually this cost is transposed to the information structuring cost which is done once with the aid of friendly data entry tools offered by the repository manager.

[3] which are links of the knowledge base.

6 Conclusion

We believe that the requirements of an advanced hypermedia system can be satisfied if it is developed over an advanced information repository. Our experience from using SIS and DOMENICUS indicates that the following benefits can be obtained from the approach :

- **Development efficiency**: Customization of ready-make functionalities, with reusability of the presentation specification. Automatic production of the card contents. Minimization of the data entry effort, usage of the same base for more than one presentations (data reuse).
- **Product quality**: Information consistency, ability to express complex queries, fully connected presentations.
- **Ease of tailoring/adaptation**: This is achieved through a simple but powerful presentation model.
- **Extensibility**: Schema and data evolution is possible at run-time.

References

[1] Philip A. Bernstein and Umeshwar Dayal. "An Overview of Repository Technology". In *Proceedings of the 20th VLDB Conference*, pages 705--713, Santiago, Chile, 1994.

[2] Panos Constantopoulos. "Cultural Documentation: The CLIO System". Technical Report 115, Institute of Computer Science Foundation for Research and Technology Hellas, January 1994.

[3] Panos Constantopoulos and Martin Doerr. "Component Classification in the Software Information Base". in O.Nierstrasz and D.Tsichritzis, eds.,Object-Oriented Software Composition, Prentice-Hall,1995.

[4] Panos Constantopoulos and Martin Doerr. "The Semantic Index System : A brief presentation". Institute of Computer Science Foundation for Research and Technology Hellas, May 1994. (http://www.ics.forth.gr/proj/isst/Systems/sis/).

[5] Hugh Davis, Wendy Hall, Ian Heath, and Gary Hill. "Towards An Integrated Information Environment With Open Hypermedia Systems". In *Proceedings of European Conference on Hypertext - ECHT '92*, Milano, Italy, November 30, 1992.

[6] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. "Telos : Representing Knowledge about Information Systems". *ACM Transactions on Information Systems*, 8(4), October 1990.

[7] John Mylopoulos and Renate Motschnig-Pitric. "Partitioning Information Bases with Contexts". In *Proceedings of Conference on Cooperative Information Systems, CoopIS-95*, pages 44--54, Vienna, Austria, May 1995.

[8] Ruben Prieto-Diaz. "Implementing Faceted Classification for Software Reuse". *Communications of the ACM*, 34(5), 1991.

[9] Manos Theodorakis. "Name Scope in Semantic Data Models". Master's thesis, Department of Computer Science - University of Crete, September 1995. (in Greek).

[10] Yannis T. Tzitzikas. "View Updates in Knowledge Bases". Master's thesis, Department of Computer Science - University of Crete, October 1995. (in Greek).

Fusing WWW and Link Server Technology: One Approach

E. James Whitehead, Jr., Roy T. Fielding, Kenneth M. Anderson

Abstract

This position paper explores how a fusion between distributed link data systems, as exemplified by the World-Wide Web (WWW), and link server systems, as exemplified by the Open Hypermedia Systems (OHS) community, might be achieved. In this hybrid system, the user's interface to their information space is presented by a network of cooperating processes (similar to both WWW client "helper" applications and OHS client applications) which interact with a local link-server process. This local link server interacts with data managers which maintain a local, personal hypertext, along with a cache of the user's local context within the global hypertext, which may be expanded by retrieving artifacts and links from HyperText Transfer Protocol (HTTP) servers. In this scenario, HTTP servers act as globally available persistent artifact databases.

Introduction

Existing approaches for providing hypermedia linking between objects with heterogeneous data types can be characterized as either "link server" systems, or "distributed link data" systems; Chimera [1], HyperForm [7], and Microcosm [4] are examples of the link server approach while the World-Wide Web (WWW) embodies the distributed link data approach. A link server provides greater control and coherency management of its hypermedia relationships and dependencies; distributed link data provides for better scalability and hyperweb robustness. This paper explores the hypothesis that the advantages of both approaches are necessary for the next generation of hypermedia systems, and gives a proposal for how these approaches might be combined. In the sections below, an overview of the link server and distributed link data approaches are given first, followed by a description of the proposed hybrid system.

Link Server Approach

The link server approach to hypermedia systems is based on cooperation between a link server and a set of clients. The clients manage the display of, and user interaction with, graphical views of resources (i.e., entities or objects) in a hyperweb; the link server manages the persistence of hypermedia information, such as links and anchors associated with each resource. A hypermedia traversal event generated by a client is translated to a query on the link server, which responds by activating the semantics associated with the traversed link. This separation of concerns allows the link server to be independent of the nature (i.e., data type) of the resources contained in a hyperweb, allows dynamic binding of link traversal semantics (actions performed by the hypermedia system in response to the link traversal), and minimizes the amount of hypermedia functionality duplicated within each client implementation.

Because in the link server approach the links and link semantics are centrally located, the link server can easily provide for management of link consistency, analysis of dependencies between resources, and a

coherent representation of the state of the hyperweb as it evolves. In addition, the separation of links from the linked resources allows anchors to be assigned without modifying the resources themselves, which is a necessity when the user does not have write-access to the resource, or when the data type of the resource has no built-in support for hypermedia anchors. It also allows links to be treated as first-class objects, in the sense that links can be grouped, analyzed, and referenced independent of the linked resources. However, the same conditions of centrality introduce problems in regard to scalability, constraints on distribution of linked resources, the inability to modify links and linked resources external to the hypermedia environment, and a single point of failure for hyperweb functionality.

Distributed Link Data Approach

The distributed link data approach used by the World-Wide Web is based on standardizing the identification of, and interfaces between, linked resources, and having the resources themselves define the link anchors and traversal semantics. Links and anchors are embedded within the standard data types, so that once a resource is retrieved, all of the information needed to perform subsequent link traversal is available. This approach emphasizes the use of a few specific data types (e.g., HTML) and access protocols (e.g., HTTP [5]), requiring the client software to perform all hypermedia functionality other than storage, and thus freeing the server for greater scalability. At the same time, distributing the link information to each resource allows each subsection of the hyperweb to be independently accessible and available for use, even when the user is isolated from the rest of the Web. Furthermore, it allows hypermedia resources to be created and modified by tools which may be completely unaware of the hypermedia environment, resulting in a pervasive distribution of information content.

The weaknesses of the distributed link data approach are in those areas where the link server approach has strength. In the WWW, data types not suited for link embedding can be the destination of hypermedia traversals, but not the initiator of a link; the information must first go through a data conversion process in order for such data to become hypermedia-aware. Likewise, the only method for analyzing links (and thus resource dependencies) in a distributed link data system is to actively traverse the hyperweb, either manually or with the assistance of software agents (e.g., web spiders). This limits the system's ability to maintain referential integrity when resources change, ability to assist the user in the navigation and visualization of the hyperweb, and ability to predict the effects of a change before it is made.

Hybrid Approach

Neither the link server approach nor the distributed link data approach are sufficient, when taken alone, to provide for the control, heterogeneity, adaptability, robustness, and unconstrained distribution necessary to support the hypermedia visualization, searching, and multi-directional linking between disparate data types of next-generation hypermedia systems.

It is our hypothesis that addressing the problems inherent in each approach requires the creation of a system which is a fusion of link server and distributed link data technology. Informing this hybrid system is the lack of scalability of the current "fatware" clients (e.g. Netscape 2.x), the increased modular extensibility within clients (e.g. Java and client-side scripting languages), and the current trend of HTTP servers towards the classic notion of an artifact database.

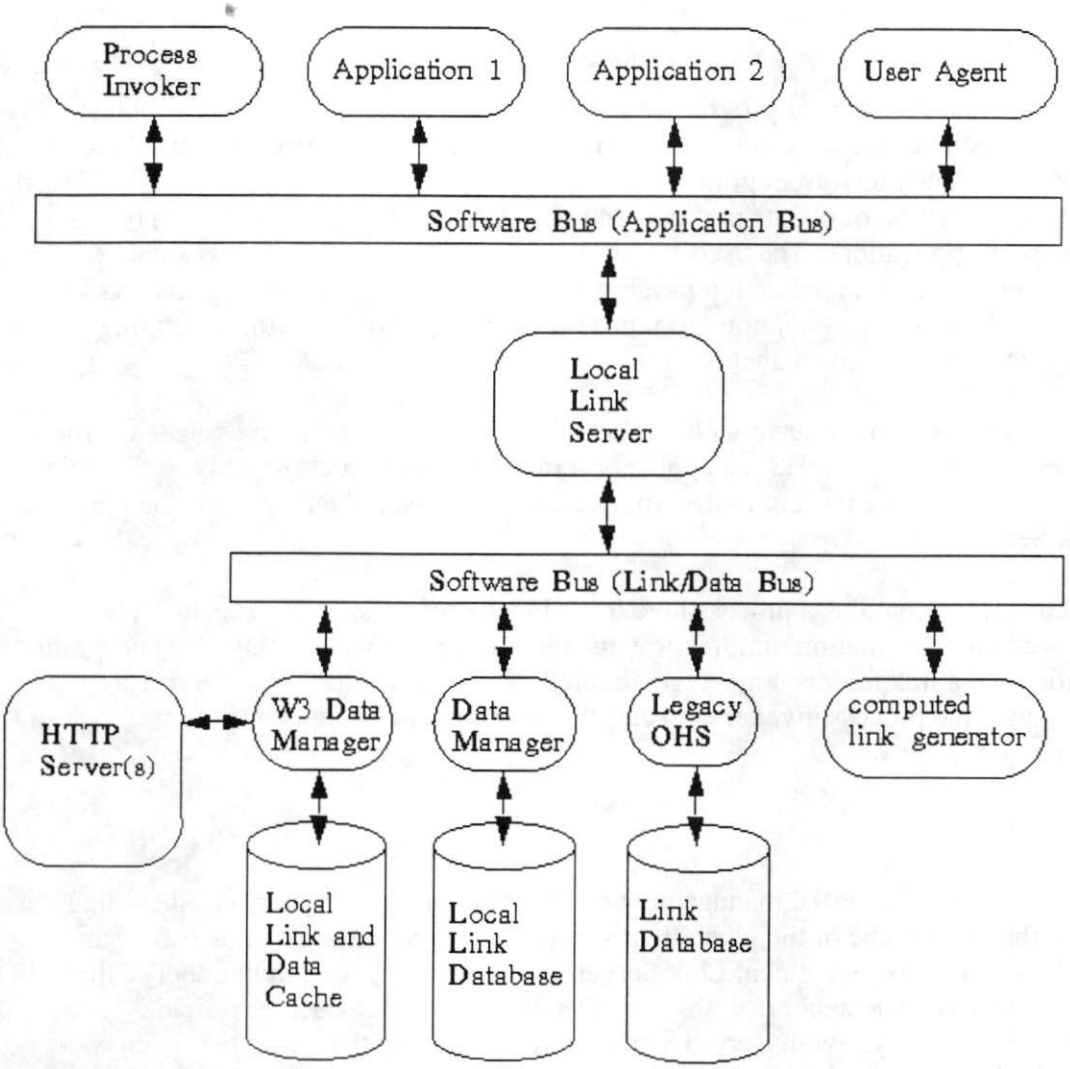


Figure 1: Proposed architecture of the combined link-server and distributed link data approach. All ovals, rounded squares, and rectangles are operating system processes. Arrows represent either communication paths or data access.

Figure 1 shows a proposed architecture for such a system, consisting of three layers, the User Application layer, the Local Link Server layer, and the Link and Data Management layer, described in the sections below. These layers are connected using a software bus.

Software bus technology was originally developed in the software environments domain, notably in commercial products such as SoftBench [3], or ToolTalk [6] (bundled with Solaris 2). The basic notion of a software bus is that it receives incoming messages, and then retransmits these messages to all interested parties. However, the message passing semantics vary for each particular bus. For this paper, we will assume our software buses may forward messages along to parties which have registered for them, and that the messages may encode either requests for future action, or notifications of prior activity. For efficiency, we also assume that large chunks of data are passed by a reference to the data chunk, rather than transmitted across the bus.

User Application Layer

The user application layer consists of viewer programs for specific data types (e.g. an MPEG player or an HTML viewer), virtual machines for distributed programs (e.g. a Java virtual machine), user agents or spiders, web visualizers -- ideally any user application program with a notion of hypertext. This layer is directly analogous to the collection of clients participating with an open hypermedia system (OHS), and to WWW client "helper" applications. The user interface for the hypermedia system is solely provided by programs in this layer, and application programs are responsible for mapping anchors and links into manipulable visualizations. Application programs are also responsible for implementing the visual behavior of a link traversal to a given anchor.

Programs in the application layer communicate with the local link server by sending messages via the Application Bus. Whereas in an existing OHS, an application might send a request directly to the link server, in this approach messages are first sent to the Application Bus, which then forwards the message along to the Local Link Server.

The User Application Layer also contains a process invoker, which is responsible for executing applications as needed when the destination of a link traversal is not currently executing. This program scans all messages waiting for a link traversal message directed to a non-executing application program. When it sees such a message, the process invoker executes the program, and retransmits the message to the now-executing application.

Local Link Server

The Local Link Server is responsible for the management of the complete local hyperweb (including the local, personal web, and the local cache of the global hypertext), and for definition of link traversal semantics across all anchors in a link. The Local Link Server also maintains a minimal model of the data objects being linked so it can associate anchors with data objects. The local link server provides programs in the User Application Layer with a consistent access interface to the multiple link databases and data repositories in the Link and Data Management layer.

Link and Data Management

The Link and Data Management layer is a collection of heterogeneous link databases and data repositories. One mandatory element of this layer is the WWW Data Manager, which contains an HTTP client interface, and maintains a local cache of links and data which represent the user's local context within the global information space. Also mandatory is the Local Data Manager, which provides local storage for user defined links between both local data items and global data items. Optionally, legacy open hypermedia systems could be added to this layer (using a wrapper, or shim), providing a means to continue using existing link bases without data conversion. Computed link managers (e.g., a dictionary link) are also possible, and would be found at this level.

Each data manager must respond to queries of the links they manage. Such queries include returning all anchors of a link (for web visualization), all links which contain an anchor (for link traversals), and the more focused queries which can result from a user search of the web (for example, all links created after a certain date). It is expected that the WWW Data Manager will need to parse the contents of HTML documents it caches to generate a link representation which can easily be queried.

All data managers listen to messages on the Link/Data Bus, waiting for a message that requires them to manipulate or query their link database or data repository. For example, assume the user has initiated a link traversal on an anchor. This causes the application to issue a link traversal message to the Application Bus, which then forwards the message along to the Local Link Server. The Local Link Server then issues a query on the Link/Data Bus for all links containing the anchor. Each data manager then queries its link database for links containing the anchor. All links which are found are then packaged into a message which is then transmitted on the Link/Data Bus, and received by the Local Link Server; these results are then translated into link traversal messages which are sent out on the Application Bus and received by user applications. If the link traversal is to an object type manipulated by the application, it then presents the link traversal to the user in an application-specific manner.

Advantages

Because the hybrid approach uses data managers which store object data and link data locally, greater control can be exercised. This allows for searching, visualization, and consistency management of the hypertext web. However, by caching portions of the global hypertext accessible via HTTP servers, this control is achieved without sacrificing the global distribution and immense information content which are the hallmarks of the WWW.

Through its use of software bus technology, the proposed hybrid architecture provides an integration framework for adding new applications, link databases, and data repositories. Since new file types are constantly being developed, this architecture provides the necessary flexibility to accommodate new, unforseen applications which manipulate them. Likewise, the HTTP standard is still under revision, and new global hypertext protocols, such as Hyper-G [2] are under development by researchers worldwide. This architecture provides the necessarily flexibility to "plug-in" new protocols as they are developed.

Conclusion

This paper has characterized link server and distributed link data approaches, finding the link server offering greater control at the cost of easy distribution and robustness; conversely, the distributed link data approach sacrifices a measure of control to gain global distribution and the ability to offer hypertext services in a rapidly reconfigured network environment. Ideally the advantages of both approaches are desirable, and we have given the sketch of a system which provides greater control over the hypertext while preserving global distribution and robustness.

References

1. Kenneth M. Anderson, Richard N. Taylor, and E. James Whitehead, Jr. Chimera: Hypertext for Heterogeneous Software Environments. In *Proceedings of ECHT'94*, Edinburgh, Scotland, September 1994, pp. 94-107.
2. K. Andrews, F. Kappe, H. Maurer, and K. Schmaranz. On Second Generation Hypermedia Systems. In *Journal of Universal Computer Science 0,0* (Pilot Issue, Nov. 1994), pp. 127-135.
3. Martin R. Cagan. The HP SoftBench Environment: An Architecture for a New Generation of Software Tools. *Hewlett-Packard Journal*, 41(3):36-47, June 1990.
4. Hugh Davis, Wendy Hall, Ian Heath, Gary Hill and Rob Wilkins. Towards An Integrated Information Environment With Open Hypermedia Systems. In *Proceedings of ECHT'92*, Milano, Italy, November, 1992, pp. 181-190.

5. R. Fielding, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol -- HTTP/1.1. Internet-Draft (work in progress), UC Irvine, MIT/LCS, November 1995.
6. ToolTalk Reference Manual, Sun Microsystems, 2550 Garcia Avenue, Mountain View, California, 94043-1100.
7. Uffe K. Wiil, and John J. Leggett. Hyperform: Using Extensibility to Develop Dynamic, Open and Distributed Hypertext Systems. In *Proceedings of ECHT'92*, Milano, Italy, November, 1992, pp. 251-261.

E. James Whitehead, Jr.
Roy T. Fielding
Kenneth M. Anderson
Information and Computer Science
University of California, Irvine CA 92717-3425

The Konstanz Hypertext System: Progress Report

Marc Rittberger
Informations Science, University of Konstanz

Introduction

The long term goal of the WITH project is the development of open hypertext systems which will allow the integration of information from various sources for diverse purposes and a wide range of applications. At the time we applied for the initial project grant, various types of commercially available databases were the main information sources which came into consideration. In the meantime developments in the context of internet especially WWW gave occasion for an integration of additional external information sources and thereby a shift in the emphasis of the project aims. As a consequence the expansion of the number of information sources has considerably increased the number of types of documents which is taken into consideration.

In this regard we developed the protoypical hypertext system KHS (Konstanz Hypertext System) which in the meantime has established itself as the core of the software work pursued in the project and serves as a basis for the experimental testing of procedures and methods developed to get a multi-resource hypertext system. Our goal is the integration of internal and external knowledge stocks, such as for example are constantly generated and managed by scientist in their workplaces, or initially processed in an electronic dossier organized as a hypertext. KHS is, however, not only a convenient interface or gateway to these services, but rather enables the appropriate hypertext processing and use of these stocks as they are embedded in a local hypertext, be it through a real or a referential integration.

To understand the KHS we will describe the basics of the system [Hammwöhner/Kuhlen 1994, Rittberger/Hammwöhner/Aßfalg/Kuhlen 1994, Aßfalg/Hammwöhner/Rittberger 1993] the knowledge based methods employed in our system, which expand type- and content-oriented filtering processes, the opening of the system in regard to the World Wide Web and show hypertext-specific search methods used in KHS.

The hypertext-model of KHS

KHS allows the integration of various application domains, the use of multiple information resources and parallel use by an - in principle - arbitrary number of users. The unifying framework is supplied by a generic, application independent hypertext-model comprising a structure model which describes the structure of well-formed hypertexts with an interaction model which defines generic interaction styles. Both the structure model and the interaction model can be refined to suit the needs of special applications or individual users.

- ❑ **The structure model**
- ❑ **The interaction model**
- ✡ **KHS Browser**

The structure model

The simple node-link structure of early hypertexts proved unable to provide sufficient orientation clues in large and complex hypertexts. KHS therefore employs additional structuring mechanisms as follows:

❖ Typing of hypertext objects allows the stepwise refinement of structure and behavior of hypertext objects. The type of a hypertext object determines its internal structure (content) and behaviour. KHS distinguishes between hypertext units, devoted to the representation of the information content of a hypertext and links, realizing the relations between such items of information.

❖ Semi-structured hypertext objects offer structured data where they are required for further inference processes.

❖ Composite nodes provide a polyhierarchical structuring mechanism. The KHS hypertext model regards composite nodes as the backbone of the hypertext structure and as a means for structured navigation. The types of units which are allowed in a composite unit are subject to type checking. If users know what kind of composite unit they have entered, they can anticipate the kind of information they will find.

The interaction model

KHS employs a multi-window interface to its hypertexts. At any given point in time user's attention is concentrated on a particular hypertext unit. The unit's content and a minimal set of contextual information (embedding in the structural hierarchy, outgoing links, etc.) are displayed within one central tool, the Hypertext Browser. Interaction with the unit takes place via a mouse click on a hotword or on lists of unit names. More complex functions can be activated by unit type-specific pop-up menus.

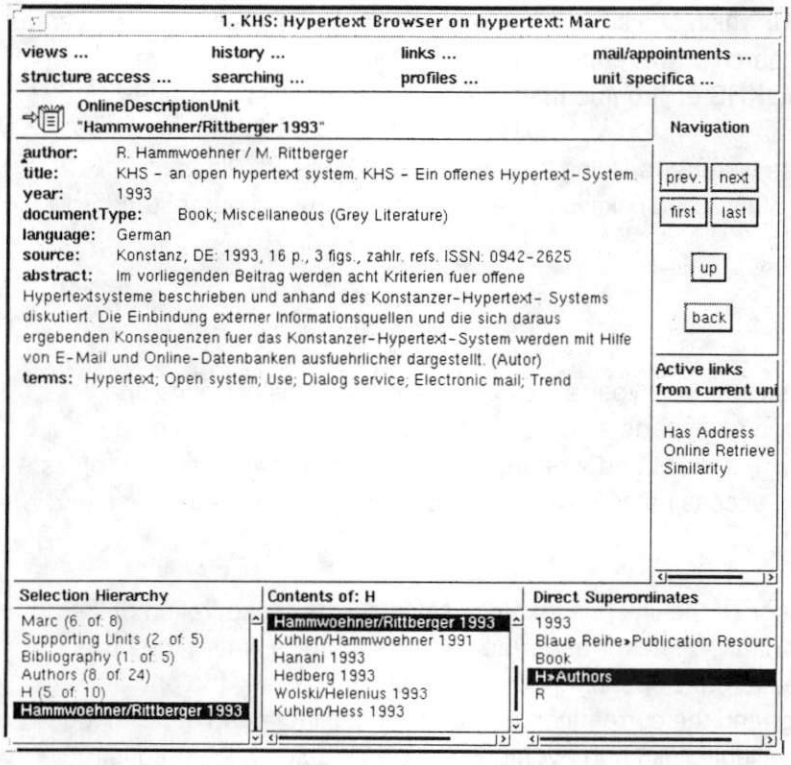
Nevertheless, no single tool can satisfy the presentation and interaction demands of a complex hypertext model. Therefore KHS provides a set of tools which can be additionally activated, providing access to:

❖ special properties of hypertext objects (units or links),
❖ the content of additional units,
❖ lists of units obtained by search processes, the dialogue history,
❖ overviews of the structural hierarchy and link webs

All of these tools communicate with each other and thus guarantee a consistent display of the actual state of the hypertext.

Working with KHS will in any case include exploratory interaction styles. When navigating through a hypertext (or a relevant subset), a user must take decisions as to which unit he regards as the most appropriate one to be read next. KHS assists him by providing as many discourse clues as possible to indicate where the next navigation step, whether it is a traversal of hierarchies, an exploration of relevance sets or link navigation.

KHS Browser



Filtering and searching in KHS

Often, especially within highly interconnected hypertexts, too much information is retrieved by a single navigation or search step. Like many other hypertext-systems, KHS offers filter options which prevent any information from being presented which does not conform to special filter conditions. The most important KHS filter types are type and structure oriented.

❖ Type based filters preclude the presentation of any units or links which do not conform to one of a set of previously chosen types.

❖ Structure based filters only regard units (and links which lead to these units) as relevant which are embedded into special branches of the multihierarchy. As these structures may be constructed dynamically and temporarily (e.g. as result of a search), these filters can be used to combine the search results of several queries.

For retrieval purpose vector based statistical analysis of the content of hypertext is available in KHS. For the selection of information sources these statistical techniques are used for clustering information units containing the description of information sources in a formal and a content based way [Rittberger 1994]. Defining starting points in the cluster, users will be able to search in an contextual environment information units relevant to their search aim.

Besides filtering KHS allows different kinds of searching in KHS hypertexts, like key-word, full text search, passage retrieval and searching with external tools in online databases or the WWW [Rittberger/Hammwöhner/Aßfalg/Kuhlen 1994] and including the result in KHS hypertexts. Besides this a search engine is going to be developed, which can search different information sources with the same tool. This tool should be able to have a search power depending to the server it is searching in, e.g. it should be more powerful in searching the KHS environment, than searching an OPAC or pages of a regular WWW-server.
Furthermore KHS will be used to evaluate whether building hypertexts with the possibilities of structuring, contextualisation, using typed units and links is more valuable than using other standard WWW-editors.

Knowledge base in KHS

KHS tries to integrate the knowledge base into the hypertext by using special types of nodes and links to represent all needed domain-knowledge as hypertext objects [Assfalg/Zink 1994]. This formal knowledge consists of the well known frames, rules and a restricted form of constraints, but also of less known forms of knowledge like tasks and access paths. Two short examples for query expansion and mail classification are given.

KHS allows context-based link presentation. If the link is a text-inclusion link, the presentation of the node content is changed too. Each user of a hypertext (reader/author) has a profile where the tasks of the user and the navigational history are stored. Depending on these tasks (especially the one the system thinks that the user is performing) and the current context (that is the history, the current node and its links, ...), the system filters the available links and eventually presents automatically constructed ones. This link filtering and creation process is controlled by rules attached to a task that controls the guidance of users.

As an open hypermedia system, KHS is connected to internet services like email. We will take a look at the mail classification, which was the first knowledge-based service in KHS. When mail comes in, the system looks (after parsing it) for a mail folder in which it can be included. It knows where to find all the mail folders of a user, and checks them for whether the mail should be included or not (a single piece of mail may be included in several mail folders). If a rule is connected to this mail folder with a special link, then this rule decides whether the mail should be included or not. This can be regarded as an approach for the knowledge-based organization and construction of (new) documents in a hypermedia system.

World Wide Web and KHS

With the acceptance of WWW through the scientific community as a publishing and information service a need for opening the KHS to the WWW exists [Aßfalg/Hammwöhner 1995] . On the one side, comparable to the integration of email and online databases a WWW-client is integrated into KHS [Bekavac 1995], available as a special unit type and usable in the common environment of the Hypertext Browser. Organization, analyzing, annotating and linking with other WWW-pages or other types of units is possible with the WWW-units as usual in KHS.

Besides this KHS allows also publishing KHS-hypertexts on the WWW [Aßfalg/Hammwöhner 1995]. Accessing KHS documents, which are handled on an object oriented database system, in principle is

possible along the Common Gateway Interface of the WWW. The HTML-Code of a KHS-hypertext object is generated by the database and along the Common Gateway Interface available all around the world with a regular WWW-client (e.g. Netscape). Besides the functionality of WWW, special KHS-features mentioned before are available for 'normal' WWW users, like navigating along the polyhierarchy, context information, unit-to-unit links, structured history, and a table of contents.

References

Aßfalg/Hammwöhner 1995

author: R. Aßfalg / R. Hammwöhner
title: Das Konstanzer Hypertext System als Teil des World Wide Web
year: 1995
citeKey: assfalg_hammwoehner1995
language: German
booktitle: Informationsmanagement in der Informationsgesellschaft. Proceedings des 2. Konstanzer Informationswissenschaftlichen Kolloquiums (KIK '95)
pages: 184-195
publisher: Universitätsverlag Konstanz
editor: P. Schieber

Aßfalg/Hammwöhner/Rittberger 1993

author: R. Aßfalg / R. Hammwöhner / M. Rittberger
title: The hypertext internet connection: E-mail, online search, gopher
year: 1993
citeKey: assfalg_etal1993
booktitle: Online Information 93. 17th International Online Information Meeting, 7.-9. December, London
pages: 453-464
publisher: Learned Information Ltd: London
editor: D.I. Raitt / B. Jeapes
issnisbn: 0-904933-85-7

Aßfalg/Zink 1994

author: R. Aßfalg / V. Zink
title: Wissensbasierte Dialogplanung für WWW am Beispiel des Konstanzer Hypertext-Systems (KHS)
year: 1994
citeKey: assfalg_zink1994
booktitle: Mehrwert von Information - Professionalisierung der Informationsarbeit. Proceedings des 4. Internationalen Symposiums für Informationswissenschaft (ISI '94)
pages: 429-438
publisher: Universitätsverlag Konstanz: Konstanz
editor: W. Rauch / F. Strohmeier / H. Hiller / C. Schlögl
volume: 16
series: Schriften zur Informationswissenschaft

Bekavac 1995

author: B. Bekavac
title: Das Konstanzer Hypertext System (KHS) als WWW-Client
year: 1995
citeKey: bekavac1995

language: German
booktitle: Informationsmanagement in der Informationsgesellschaft. Proceedings des 2. Konstanzer Informationswissenschaftlichen Kolloquiums (KIK '95)
pages: 196-209
publisher: Universitätsverlag Konstanz
editor: P. Schieber

Hammwöhner/Kuhlen 1994

author: R. Hammwöhner / R. Kuhlen
title: Semantic control of open hypertext systems by typed objects
year: 1994
citeKey: hammwoehner_kuhlen1994
journal: Journal of Information Science
volume: 20
number: 3
pages: 175-184

Rittberger 1994

author: M. Rittberger
title: Support of online database selection in KHS
year: 1994
citeKey: rittberger1994
booktitle: National Online Meeting'94, New York 10 -12 May
pages: 379-387
editor: M.E. Williams

Rittberger/Hammwöhner/Aßfalg/Kuhlen 1994

author: M. Rittberger / R. Hammwöhner / R. Aßfalg / R. Kuhlen
title: A homogenous interaction platform for navigation and search in and from open hypertext systems
year: 1994
citeKey: rittberger_etal1994
booktitle: RIAO 94 Conference Proceedings. Intelligent multimedia information retrieval systems and management
pages: 649-663
organization: Rockefeller University
address: New York, NY - USA October 11-13

Workshop Participant Addresses

Ajit Bapat
GMD-IPSI
Dolivostr. 15
D-64293 Darmstadt
Germany
bapat@darmstadt.gmd.de

Hugh C. Davis
Dept. of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ
England
H.C.Davis@ecs.soton.ac.uk

Thomas Dedonno
Dept. of ECE - 0408
University of California, San Diego
La Jolla, CA 92093-04088
USA
tdedonno@sdcc3.ucsd.edu

Serge Demeyer
Programming Technology Lab
Brussels Free University
Pleinlaan 2, B-1050 Brussels
Belgium
sademeye@vnet3.vub.ac.be

Kaj Grønbæk
Computer Science Dept.
Aarhus University
Ny Munkegade 116, DK-8000 Aarhus C
Denmark
kgronbak@daimi.aau.dk

Jörg M. Haake
GMD-IPSI
Dolivostr. 15
D-64293 Darmstadt
Germany
haake@darmstadt.gmd.de

Wendy Hall
Dept. of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ
England
W.Hall@ecs.soton.ac.uk

Peter J. Nürnberg
Dept. of Computer Science
Texas A&M University
College Station, Texas 77843-3112
USA
pnuern@bush.cs.tamu.edu

Steven E. Poltrock
Boeing Computer Services
P.O. Box 24346, MS 7L-64
Seattle, WA 98124-0346
USA
poltrock@atc.boeing.com

Siegfried Reich
Dept. of Information Systems
University of Linz
Altenbergerstr. 69, A-4040 Linz
Austria
reich@ifs.uni-linz.ac.at

Marc Rittberger
Dept. of Information Science
University of Konstanz
PO Box 5560, D-78434 Konstanz
Germany
Marc.Rittberger@uni-konstanz.de

John B. Smith
Dept. of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175
USA
jbs@cs.unc.edu

Li-Cheng Tai
Dept. of ECE - 0407
University of California, San Diego
La Jolla, CA 92093-0407
USA
atai@ece.ucsd.edu

Richard N. Taylor
Dept. of Information and Computer Science
University of California, Irvine
Irvine, CA 92717-3425
USA
taylor@ics.uci.edu

Manos Theodorakis
Institute of Computer Science
Foundation for Research and Technology - Hellas
Science and Technology Park of Crete
Vassilika Vouton, P.O. Box 1385
GR 711 10 Heraklion, Crete, Greece
etheodor@ics.forth.gr

Randall H. Trigg
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
USA
trigg@parc.xerox.com

Yannis Tzitzikas
Institute of Computer Science
Foundation for Research and Technology - Hellas
Science and Technology Park of Crete
Vassilika Vouton, P.O. Box 1385
GR 711 10 Heraklion, Crete, Greece
tzitzik@ics.forth.gr

E. James Whitehead, Jr.
Dept. of Information and Computer Science
University of California, Irvine
Irvine, CA 92717-3425
USA
ejw@ics.uci.edu

Uffe Kock Wiil
Dept. of Computer Science
Aalborg University
Fr. Bajers Vej 7E, DK-9220 Aalborg Ø
Denmark
kock@iesd.auc.dk

Kasper Østerbye
Dept. of Computer Science
Aalborg University
Fr. Bajers Vej 7E, DK-9220 Aalborg Ø
Denmark
kasper@iesd.auc.dk