Proceedings of the Workshop on Virtual Documents, Hypertext Functionality and the Web

Eighth International World Wide Web Conference
Tuesday May 11, 1999 - Toronto, Canada

Maria Milosavljevic
Fabio Vitali
Carolyn Watters (eds.)

Technical Report UBLCS-99-10

May 1999

Department of Computer Science
University of Bologna
Mura Anteo Zamboni, 7
40127 Bologna (Italy)

The University of Bologna Laboratory for Computer Science Research Technical Reports are available via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` in compressed PostScript format. Abstracts are available from the same host in the directory `/pub/TR/UBLCS/ABSTRACTS` in plain text format. All local authors can be reached via e-mail at the address *last-name*`@cs.unibo.it`.

Recent titles from the UBLCS Technical Report Series

97-6 *A Process Algebraic View of Linda Coordination Primitives*, N. Busi, R. Gorrieri, G. Zavattaro, May 1997.

97-7 *Validating a Software Architecture with respect to an Architectural Style*, P. Ciancarini, W. Penzo, July 1997.

97-8 *System Support for Partition-Aware Network Applications*, Ö. Babaoğ lu, R. Davoli, A. Montresor, R. Segala, October 1997.

97-9 *Generalized Semi-Markovian Process Algebra*, M. Bravetti, M. Bernardo, R. Gorrieri, October 1997.

98-1 *Group Communication in Partitionable Systems: Specification and Algorithms*, Ö. Babaoglu, R. Davoli, A. Montresor, April 1998.

98-2 *A Catalog of Architectural Styles for Mobility*, P. Ciancarini, C. Mascolo, April 1998.

98-3 *Comparing Three Semantics for Linda-like Languages*, N. Busi, R. Gorrieri, G. Zavattaro, May 1998.

98-4 *Design and Experimental Evaluation of an Adaptive Playout Delay Control Mechanism for Packetized Audio for use over the Internet*, M. Roccetti, V. Ghini, P. Salomoni, M.E. Bonfigli, G. Pau, May 1998 (Revised November 1998).

98-5 *Analysis of MetaRing: a Real-Time Protocol for Metropolitan Area Network*, M. Conti, L. Donatiello, M. Furini, May 1998.

98-6 *GSMPA: A Core Calculus With Generally Distributed Durations*, M. Bravetti, M. Bernardo, R. Gorrieri, June 1998.

98-7 *A Communication Architecture for Critical Distributed Multimedia Applications: Design, Implementation, and Evaluation*, F. Panzieri, M. Roccetti, June 1998.

98-8 *Formal Specification of Performance Measures for Process Algebra Models of Concurrent Systems*, M. Bernardo, June 1998.

98-9 *Formal Performance Modeling and Evaluation of an Adaptive Mechanism for Packetized Audio over the Internet*, M. Bernardo, R. Gorrieri, M. Roccetti, June 1998.

98-10 *Value Passing in Stochastically Timed Process Algebras: A Symbolic Approach based on Lookahead*, M. Bernardo, June 1998.

98-11 *Structuring Sub-Populations in Parallel Genetic Algorithms for MPP*, R. Gaioni, R. Davoli, June 1998.

98-12 *The Jgroup Reliable Distributed Object Model*, A. Montresor, December 1998 (Revised March 1999).

99-1 *Deciding and Axiomatizing ST Bisimulation for a Process Algebra with Recursion and Action Refinement*, M. Bravetti, R. Gorrieri, February 1999.

99-2 *A Theory of Efficiency for Markovian Processes*, M. Bernardo, W.R. Cleaveland, February 1999.

99-3 *A Reliable Registry for the Jgroup Distributed Object Model*, A. Montresor, March 1999.

99-4 *Comparing the QoS of Internet Audio Mechanisms via Formal Methods*, A. Aldini, M. Bernardo, R. Gorrieri, M. Roccetti, March 1999.

99-5 *Group-Enhanced Remote Method Invocations*, A. Montresor, R. Davoli, Ö. Babaoğlu, April 1999.

99-6 *Managing Complex Documents Over the WWW: a Case Study for XML*, P. Ciancarini, F. Vitali, C. Mascolo, April 1999.

99-7 *Data-Flow Hard Real-Time Programs: Scheduling Processors and Communication Channels* in *a Distributed Environment*, R. Davoli, F. Tamburini, April 1999.

99-8 T*he MPS Computer System Simulator*, M. Morsiani, R. Davoli, April 1999.

99-9 *Action Refinement*, R. Gorrieri, A. Rensink, April 1999.

99-10 *Proceedings of the Workshop on Virtual Documents, Hypertext Functionality and the Web*, M. Milosavljevic, F. Vitali, C. Watters (eds.), May 1999.

Table of content

Introduction

Maria Milosavljevic[1], Fabio Vitali[2], and Carolyn Watters[3]

[1] CSIRO Mathematical and Information Sciences, Australia
[2] University of Bologna, Italy
[3] Dalhousie University, Canada

This volume contains the submissions to the Workshop on Virtual Documents, Hypertext Functionality and the Web, held on May 11, 1999 at the Eighth International World Wide Web Conference (WWW8) in Toronto, Canada.

This workshop was born as the confluence of two previous series of workshops on related topics: the Hypertext Functionalities Workshop series (of which this is the eighth in the series), and the Reuse of Web Information/Flexible Hypertext Workshop series. Fruitful discussions among the organizers of the two parallel workshops at the seventh International World Wide Web Conference (WWW7) in Brisbane, Australia, April 1998, heightened an awareness of the similarities in the topics and the potential synergy of a combined workshop.

By "hypertext functionality" we mean much more than browsing by clicking on "goto" links from one node to another. The focus of the HTF series is on the identification of characteristics that define and describe the "hypertextuality" of software systems. For instance, it aims at describing new ways to view a system's knowledge and processes from a conceptual point of view, to let users access and navigate through the items of interest, to enhance the system's knowledge through comments and relationships, and to customize information and display to the individual users and their tasks. This research is being brought to the forefront within the context of the World Wide Web, and specifically related to virtual documents, as this provides an additional layer of complexity to the issues.

"Virtual documents" are web documents for which the content, nodes or links, or all three, are created as needed. There already exist several kinds of virtual documents on the web for which the content is determined dynamically. First, a template can be used for which node contents are substituted at runtime. Second, applications, like Maple or Mathematica, can be used to generate values for one time use. Third, CGI scripts and search engines can be used to compose virtual documents from fragments of other documents for the user on demand. Fourth, metadata can be generated for summarization for users, where the extraction and summarization is done on the fly for the user. Finally, natural language generation techniques can be employed to dynamically construct virtual documents from underlying data in data or knowledge bases.

The eight short papers collected here provide an overview of related research and reflections on the convergence of hypertext functionality and virtual documents on the world wide web. We will give here a brief introduction to these works:

At an abstract level, Watters and Shepherd provide a list of fundamental, systemic issues to be identified when discussing the production and use of virtual documents in the world wide web, and Crowston and Williams discuss the value of genres in Web documents, structures that are also important even in a new medium such as the Web. At a development level, Green, Milosavljevic, Dale and Paris report on their actual experiences with the development of systems generating web documents on the fly, especially with the aid of natural language processors.

Providing formal grounds to the issue of dynamically creating virtual documents, Azevedo de Lima and Price propose a methodology using design patterns, which contrasts nicely with the model of Ranwez and Crampes based on composable Information Bricks.

At a methodological level, several papers identify the atomic blocks of virtual documents and discuss how to enrich them with the information needed to generate different virtual

documents. Martin and Eklund propose to embed machine-understandable information (conceptual commands) in the source, real documents. Rock, Cawsey, McAndrew and Bental rely on associating documents with metadata described using standard metainformation sets. Finally Caumanns proposes an architectural framework for software modules to create, collect and compose chunks of inforation into virtual documents.

We believe these presentations and the discussion following in the workshop may help in answering important questions related to virtual documents and virtual application domains. How are virtual documents defined and managed? The management of this class of documents requires new understandings of bookmarking, versioning, authentication, structure, ownership, navigation, collaboration, and reuse of components. Issues of security, data protection, verification, and access control need to be addressed. Finally we need to address questions about how to determine if these web information systems are actually improving service to the users.

Of course, this workshop is part of a growing body of connected events on similar topics, which is worthwhile to list here:

- The Hypertext Functionality (HTF) workshop (the collected proceedings can be found at http://www.cs.nott.ac.uk/~hla/HTF/HTF-workshops.html) began in conjunction with the ACM Hypertext Conferences. The first three HTF workshops concentrated on the identification and organization of hypertext functionalities that could form the core of hypertext systems in a wide variety of application areas. HTF4 examined issues related to the incorporation of advanced hypertext functionality in web-based applications. HTF5, held in conjunction with the ICSE conference in Kyoto, May 1998, examined the impact of HTF on software engineering.
- The 2nd Workshop on Adaptive Hypertext and Hypermedia (http://wwwis.win.tue.nl/ah98/), held in conjunction with the Ninth ACM Conference on Hypertext and Hypermedia.
- The Workshop on Reuse of Web-based Information and 2nd Flexible Hypertext Workshop (http://www.mel.dit.csiro.au/~vercous/REUSE/WWW7-reuse.html), held in conjunction with the 7th International World Wide Web Conference (WWW7),
- The Flexible Hypertext Workshop (http://www.mri.mq.edu.au/~mariam/flexht/), held at the Eighth ACM International Hypertext Conference (Hypertext'97),
- The Intelligent educational systems on the World-Wide Web (http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/Proceedings.html), held in conjunction with the 8th World Conference on Artificial Intelligence in Education (AI-ED 97)
- The Workshop on Adaptive Systems and User Modeling on the World Wide Web (http://zaphod.cs.uni-sb.de/~UM97/ws5.html), held in conjunction with the Sixth International Conference on User Modeling (UM'97)
- The Workshop on User Modelling for Information Filtering on the World Wide Web (http://www.cs.su.oz.au/~bob/um96-workshop.html), held in conjunction with the Fifth International Conference on User Modeling (UM'96)
- The Workshop on Adaptive Hypertext and Hypermedia (http://www.education.uts.edu.au/projects/ah/AH-94.html), held in conjunction with the Fourth International Conference on User Modeling (UM'94).

Research Issues for Virtual Documents

Carolyn Watters and Michael Shepherd

Faculty of Computer Science
Dalhousie University, Halifax, NS, Canada B3J 2X4

An electronic document consists of both the content and the links associated with that document. Therefore, documents on the Web may be composed of one or more Web pages [Crowston & Williams, 1999]. Such documents may be static and persistent or they may be generated dynamically and be virtual. A virtual document is a document for which no persistent state exists and for which some or all of each instance is generated at run time [Watters, 1999]. A virtual document can then be multiple pages, a guided tour, Java applets, or application results, and may or may not have associated links. The content may be defined by tags, a template, a program, a database query, or by some application. Virtual documents have grown out of a need for interactivity and individualization of documents, particularly on the web.

The paradigm of the Web has shifted our expectations for access to information. Previously, we accessed information by the retrieval of electronic copies of documents from a large repository of relatively static information. We now expect to access information through the *manipulation* of a large collection of information resources. Some of these resources are documents and some of these resources are processes that create documents. In addition, the role of user is shifting from reader to active participant and author. Users expect hypertext functionality to be available with digital documents, i.e., users expect to be able to make comments and annotations, to be able to initiate discussion, and to be able to add content and links while reading, both individually and collaboratively.

Research Issues

A number of interesting research issues must be resolved surrounding these virtual documents on the Web. These issues cover a wide range and are described briefly below.

Generation - At what point in time is a virtual document defined? A virtual document can be defined by an author through the use of templates and links or it can be defined as the result of a search or application. Guided tours can be generated dynamically, based on an information need as defined by a user profile and / or an explicitly stated query.

Search - How do you search for virtual documents? What is the domain in which to perform the search? Will the document exist by the time the user requests it?

Revisiting - Users have an expectation that documents found once will be available on a subsequent search. The notion of *bookmark* does not apply to virtual documents in its normal, simplistic way. Bookmarks need enough information to recreate the document as it was.

Versioning - Version control has long been a concern of Information Retrieval research and is now a central issue for management of virtual documents. Users need to be able to return to a bookmarked version of a virtual document and to go forward and backward in time through changes to that virtual document.

Authentication - Who is responsible for the quality of the contents of a virtual document where components may come from a variety of sources and / or processes?

Reference - How do authors cite virtual documents or versions of virtual documents?

Annotation - The roles of user of information and supplier of information are merging. Readers expect to be able to add data, such as, comments, annotations, paths, and links, as well as content, while they are reading.

Summary

The web has not only increased the scale of information retrieval systems and applications but has also introduced a new variation of the notion of document. Basic research is required to provide the same level of understanding and measures of effectiveness and efficiency of access to virtual documents as has been achieved for persistent documents.

References

Crowston, K. and M. Williams. 1999. The Effects of Linking on Genres of Web Documents. *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*. Maui, Hawaii. CD-ROM Publication

Watters, C. 1999. Information Retrieval and the Virtual Document. *Journal of the American Society for Information*. To appear. Hawaii International Conference on System Sciences. Maui, Hawaii. CD-ROM Publication

Watters, C. 1999. Information Retrieval and the Virtual Document. *Journal of the American Society for Information*. To appear.

Navigational Context Design Pattern: An Implementation for Web Development

Flávio Azevedo de Lima, B.Sc., and R. T. Price, Eng., M.Sc., D.Phil

Amadeus Project, sponsored by CNPq and FAPERGS
Instituto de Informática, UFRGS, Brazil
(flima | tomprice@ inf.ufrgs.br)

Abstract

This paper presents an implementation of the Navigational Context hypermedia design pattern. This pattern has been proposed by Rossi, Schwabe and Garrido, but implementation alternatives have not been fully explored specially regarding Web applications. The approach presented in this paper employs features of the HTML language to implements the features specified by the design pattern.

KEYWORDS: Design Patterns, Hypermedia Design, HTML, Web Development.

1. Introduction

The main benefit of design patterns is the reuse of the design experience and / or structures [1, 2]. A design pattern catalogue has been proposed by Gamma et al. [1], and, many new patterns have been proposed [8, 9], including recent efforts towards patterns for hypermedia [2, 8]. This paper presents a new implementation approach to the Navigational Context hypermedia design pattern [2] suited for the characteristics and tools available for the Internet environment.

1.1 The Navigational Context Pattern

The Navigational Context pattern is based on the Decorator [1] (see figure 1 below). The goal of the Decorator pattern is to dynamically attach additional responsibilities to an object, in a flexible way not provided by sub-classing. In other words, Decorator detaches the object behaviour from some (or all) of the exhibition issues. By using the decorator pattern an object may dynamically receive additional features, depending on the context where it is being used, without changing its core features.

The Navigational Context Pattern benefits from the experience achieved from OOHDM [3]. The use of this pattern for hypermedia purposes, as presented in [2], employs characteristics of the Decorator pattern to include specific information about the context where a node is being displayed. It simplifies the modelling of the navigational structure of the application, because the access to information nodes is specified within the context. It also helps users not getting lost, because generic information regarding the context is displayed within the context node. However, the pattern description [2] is concerned only with the implementation on hypermedia environments, and explicitly states that the implementation for Web development may be difficult.

2 Web development and document management

One of the research areas of the Amadeus Project has been the modelling of Web Information Systems based on document management and workflow [4]. The goal is to achieve an integrated modelling methodology suited for the particular characteristics of the Web, involving documents, executable components, workflow, web browsers, and so on. To do so, a number of UML stereotypes [5] have been defined (for instance, «Document», «Context», «Executable Component», «Frame», and «Navigational Link») and modelling steps have been suggested. One of the challenges faced was the specification of access

constraints in terms of roles played by different actors. Access constraints specify the concrete views of a document available for each actor. In the modelling methodology currently in development within the Amadeus Project it has been realised that this problem may be solved by the definition of contexts for actor and documents. Contexts are stereotypes that define how a document is shown. The relationships (accesses) among actors and documents are not direct; there is always a context between an actor and a document. The context is responsible for displaying the appropriate information contained in a document.

The definition of how a context relates to a document is modelled in a specific step of the methodology. Other important phases are uses cases, global navigational definition, structure of each document and data access constraints with contexts. Documents are often a composite of subdocuments with specific sets of information. For each root document, a number of contexts may be defined, depending on different access restrictions for each actor in the system. Each context must be modelled separately, specifying which and how subdocuments and smaller pieces of information will be displayed. A consequence of this approach is that a document is never responsible for displaying itself and its subdocuments.

2.1 Navigational Context extended

The combination of features from hypermedia development and document management for Web systems results in an extended design pattern with a particular implementation that:

- uncouples the navigational objects from the context in which they are to be displayed;
- groups different objects related to each other;
- adds specific information related to the context;
- creates a navigational structure that helps users not getting lost;
- uncouples the roles of model (information) and view (display) from a node;
- specifies which information is available for each participant of the system;
- provides an open, standard-based implementation suitable for Web development.

3. HTML Features

This section briefly introduces some useful features of HTML that make possible the implementation of the concepts presented in the paper. The more important features are the Document Object Model (DOM, [6], [7]) and Cascading Style Sheets (CSS, [6]), in addition to JavaScript [6]. These tools allow a complete, dynamic manipulation of an HTML page, providing the ability to modify document for presentation in different contexts.

DOM is a great advance in HTML development. It combines HTML elements (treated, now, as objects), CSS, and script languages to manipulate both. DOM is an API for HTML and XML [6] documents. It specifies the attributes and methods of any element or tag found in an HTML or XML document. DOM also defines the logical structure of documents and the way a document is accessed and manipulated. DOM is an "object model" in the traditional object oriented design sense. Documents are modelled using objects, and the model encompasses not only the structure of a document, but also the behaviour of a document and the objects of which it is composed. JavaScript adds dynamic behaviour, manipulating the document elements defined by the DOM.

On the other hand, CCS lets the developer separate the content of an HTML document from their presentation. It defines, in an independent manner, how different HTML elements the browser will display. Therefore, it's possible to create a unique CSS with global rules that must be used by all documents in a site. For example, it's possible to decide that every <H1> must be displayed in blue by defining a rule like "H1 {color: blue}" within a style sheet. It's also possible to define styles for specific elements and declare

element classes, among other possibilities.

4. Pattern Implementation

The Decorator pattern has the following structure [1]:

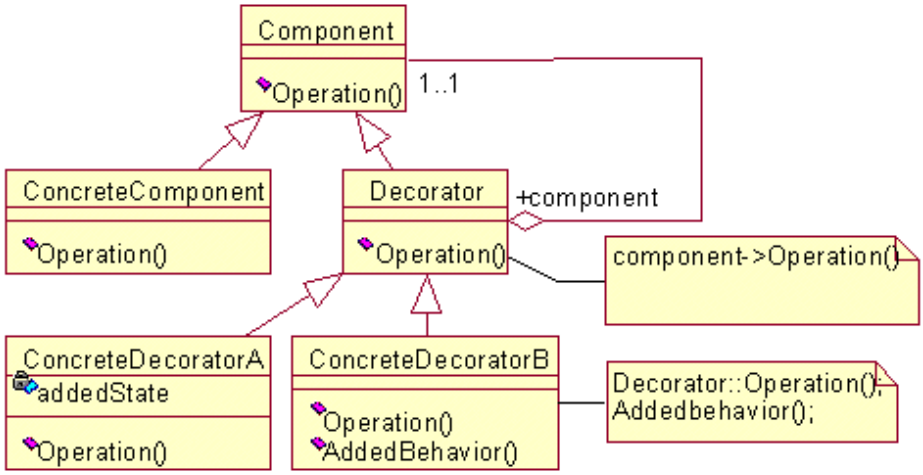


Figure 1: Decorator design pattern

ConcreteDecorators aggregate components, which may be another Decorator or a ConcreteComponent. The implementation here presented translates this structure into a simpler composition, where a document (a ConcreteComponent) may be directly aggregated by contexts (a ConcreteDecorator). There is no real need for the abstract classes, because HTML files do not have all the formal semantics of programming classes.

The extended Navigational Context, modelled with an UML stereotype based on Class, is an aggregation of HTML Frames, another stereotype. Each frame of the context is responsible by showing one subdocument, specifying which information will be displayed. The context may (and often do) provide additional frames or fields to show specific information and links. Figure 2 shows a Navigational Context definition, which has been implemented for experimentation. The notation used is the most recent outcome of the Amadeus Project, evolved from a previous work [4], and is explained below.

The context "SummContext" is responsible by the navigation through a set of Staff pages, either teachers or students (this choice is specified in the "Index" document). The function of the context is to hide some of the information contained by the actual document. Two frames ("SummControlFrame" and "SummDisplayFrame") compose the context. "SummDisplayFrame" exhibits the "Staff" document. "SummControlFrame" implements additional functionality, providing links to other documents of the same type (teacher or student, as chosen), a link to the home page, and a description of the current context. The "SummScript" executable component contains the JavaScript code that manipulates the document. To specify how the context affects the document, every attribute and method of the document must be duplicated in the respective frame, with the appropriate changes in permissions. "SummDisplayFrame" shows an example: in the document, all attributes are invisible; in the frame, "name" and "email" are visible, but locked (this is a new convention not supported by Rational Rose, the tool currently in use).

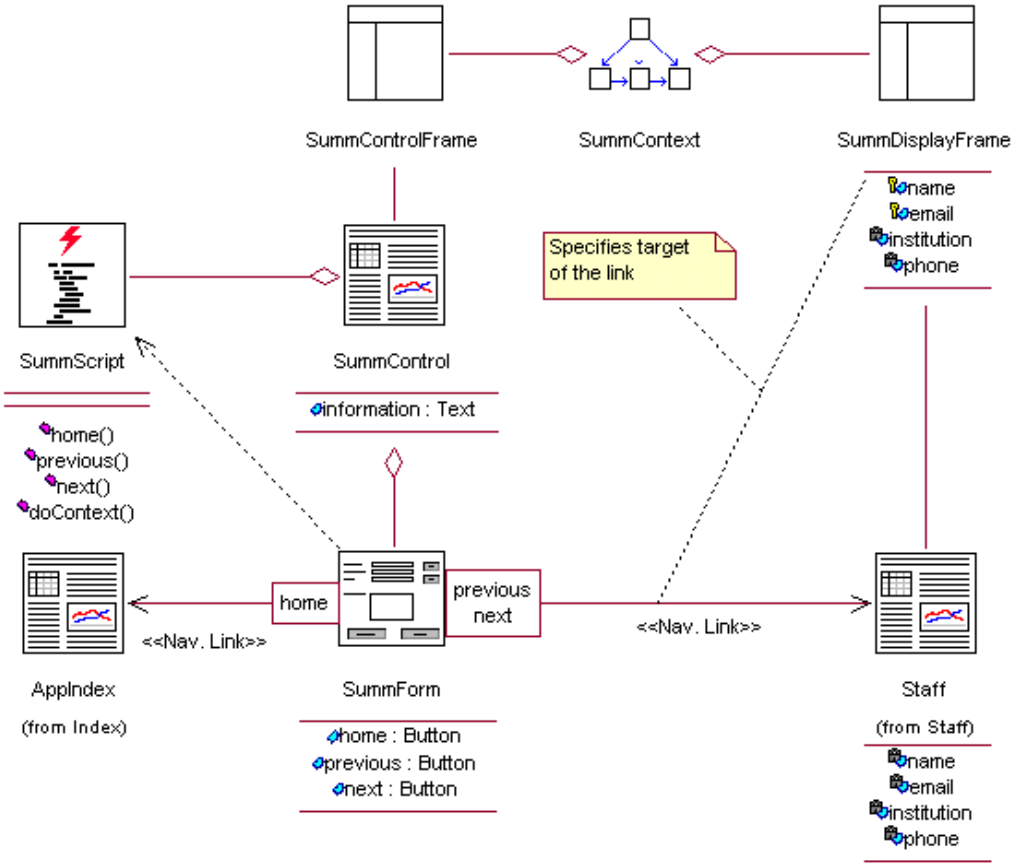


Figure 2: "Summarised Information" Navigational Context

To use the context, the user must specify the type of the staff in a previous page. When the context is loaded, it is responsible by retrieving the appropriate set of documents (in this example, the documents are hard coded in the script). Specific information and links are provided in an additional frame.

One of the advantages of this approach is the easy translation of the visual model to implementation HTML files and JavaScript code, because model elements are near to implementation elements. The example implementation contains the following HTML files:

- **_GlobalIndex**: frameset that contains "_GlobalHidden" and the main page of the application, "AppIndex". This is the page referenced by the user to load the application;
- **_GlobalHidden**: hidden document that contains global JavaScript code and variables to pass information among documents;
- **AppIndex**: the main page of the application;
- **_Null**: blank page for implementation issues;
- **SummContext**: frameset that contains the frames "SummControlFrame" and "SummDisplayFrame". This is the actual context, the page referenced by the "AppIndex" page to load the documents within the context;
- **SummControl**: contains additional information of the context (contextual links and a text description) and the JavaScript code that manipulates the "Staff" document. The functions in "SummScript" displays the appropriate attributes of the document;

The influence of the context over the documents is achieved with the use of the HTML features described before. The sequence of actions is:

1. User loads "_GlobalIndex.html"
   1.1. Document "_GlobalHidden.html" is loaded in a hidden frame

1.2. Document "AppIndex.html" is loaded in the main frame

2. User selects the radio button "Teachers" in "AppIndex.html", for example
3. User clicks on the button "Summary" in "AppIndex.html"

3.1. The function "onSumm( )" in "AppIndex.html" is executed

3.1.1. Sets a global variable in "_GlobalHidden.html" to "teachers"

3.1.2. Sets a global variable in "_GlobalHidden.html" to "summary"

3.1.3. Calls function "loadContext( )" in "_GlobalHidden.html"

3.1.3.1. Loads "**SummaryContext.html**" in the main frame

3.1.3.1.1. Loads "_Null.html" in the "display" frame

3.1.3.1.2. Loads "SummControl.html" in the "control" frame

3.1.3.1.2.1. Loads first teacher document in the "display" frame

3.1.3.1.2.1.1. Calls "doContext()" in "SummControl.html"

3.1.3.1.2.1.1.1. Shows the appropriate attributes of the document

While modelling the document, the designer must be aware of the contexts in which the document will be displayed to decompose the document into appropriate subdocuments. Each subdocument must also be designed with the contexts in mind, to achieve an adequate modelling of its elements.

Although the example is simple, it makes clear the great opportunities open by the use of the DOM and script languages for the implementation of Navigational Contexts. For example, the type of user may be identified in a login page and then appropriate contexts may be automatically chosen.

5. Conclusion

An alternative implementation for the Navigational Context has been presented. It uses advanced characteristics of HTML to implement the features defined by the pattern. The additional features obtained from this approach

- provides an open, standard-based implementation suitable for Web development;
- provides a simple strategy to automatically translate visual models to HTML files;
- provides a very powerful way to manipulate every element in HTML documents;
- uncouples the roles of model (information) and view (display) from a node. A document becomes unaware of its presentation;
- formally specifies which information is available for each participant of the system at design time;
- requires the use of a context to access a document (at least, a default, transparent context must be provided);
- requires a way to pass information about documents across different frames. A context must receive parameters to retrieve the appropriate documents. It can be achieved by the use of global JavaScript variables in hidden frames.

This work is currently under development within the scope of the Amadeus Project and will be part of a master's thesis.

References

[1] E. Gamma; R. Helm; R. Johnson; J. Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Reading: Addison Wesley Longman, 1995.

[2] G. Rossi; D. Schwabe; A. Garrido. *Design Reuse In Hypermedia Applications Development*. Proceedings of the Eight ACM International Conference on Hypertext, Southampton, 1997.

[3] D. Schwabe; G. Rossi; S. Barbosa. *Systematic Hypermedia Design With OOHDM*. Proceedings of the Seventh ACM International Conference on Hypertext, Washington, 1996.

[4] F. A. de Lima; R. T. Price. *Towards an Integrated Design Methodology for Internet-based Information Systems*. Fifth International Workshop on Engineering Hypertext Functionality, in conjunction with the International Conference on Software Engineering. Kyoto, 1998. Available on WWW at http://www.ics.uci.edu/~kanderso/htf5/papers/flima/.

[5] H-E Eriksson; M. Penker. *UML Toolkit*. New York: Wiley Computer Publishing, 1998.

[6] E. Holzschlag. *Special Edition Using HTML 4, Fifth Edition*. Que Education & Training, 1998.

[7] *Document Object Model (DOM) Level 1 Specification*, W3C Recommendation 1 October, 1998. Available on WWW at http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/.

[8] C. Schmidt (editor). *Pattern Languages of Program Design*. Addison-Wesley, 1995.

[9] C. Larman. *Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design*. Prentice-Hall PTR, 1997.

The MIRADOR project

Metadata Improved Relevance Assessment through Descriptions of Online Resources

Sheila Rock, Alison Cawsey, Patrick McAndrew and Diana Bental

Department of Computing and Electrical Engineering
Heriot-Watt University
Edinburgh EH14 4AS, UK
mirador@cee.hw.ac.uk

Abstract

MIRADOR (Metadata Improved Relevance Assessment through Descriptions of Online Resources) is concerned with using metadata to produce tailored descriptions of online resources, that will help people evaluate their relevance and suitability. In this document, we describe the background to MIRADOR, and then discuss some early evaluative work that we have begun.

1. Introduction

There is a great diversity of multimedia resources available on the Internet. Such diversity and volume is very useful in most contexts, and educational ones, which are the focus of our investigation, are no exception. However, as the volume of resources increases, it becomes difficult to track down material relevant to an individual, and expensive to download a resource just to see if it is relevant.

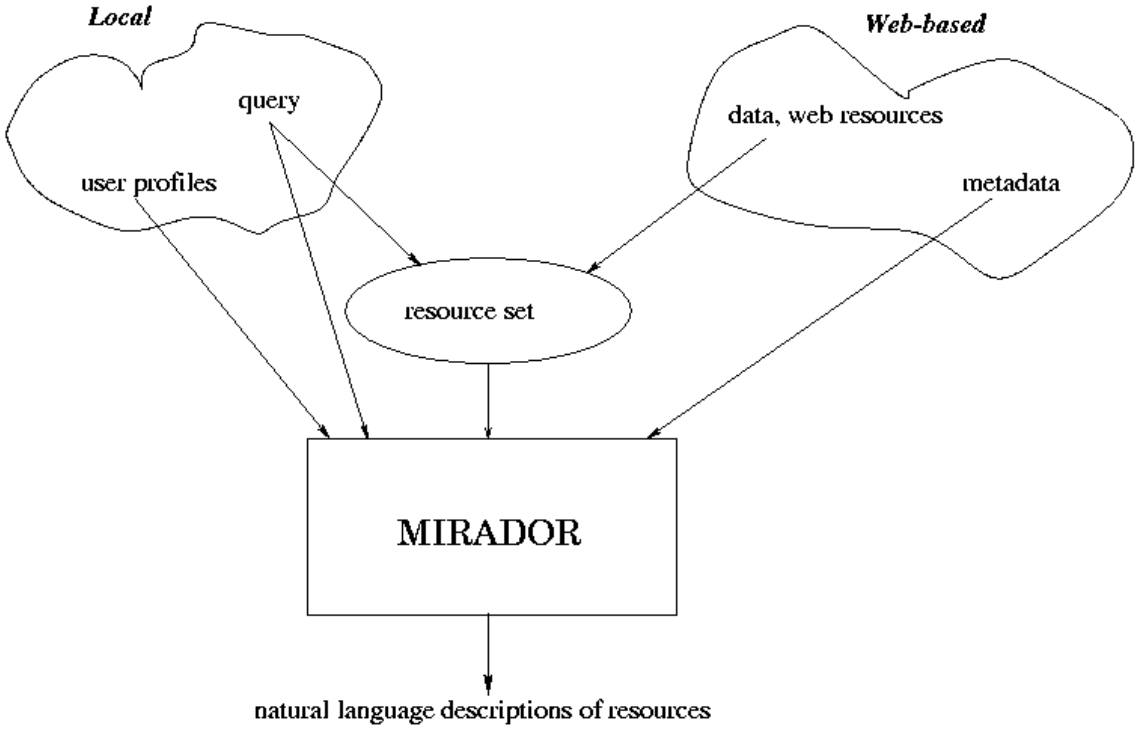


Figure 1 An overview of the MIRADOR approach

The main response to this problem has been in the form of better search and retrieval tools.

However, we believe it is also beneficial to approach this issue from the user end (in the educational context, users may be teachers, learners, or anyone making use of a resource in any way), by providing mechanisms whereby the user is more able to assess the potential relevance of a resource of interest. In particular, providing the user with better descriptions of the resources will aid this assessment.

Our aim is therefore to support the user in searching for multimedia resources on the web by:

- providing tailored descriptions of existing networked resources
- taking into account user profiles and richer resource descriptions
- using metadata to generate these tailored descriptions.

Figure 1 is an overview of our approach. We distinguish between local information, such as user profiles and the particular query that a user might have, and web-based information, which includes the web resources themselves and the metadata associated with them. In a conventional search, the query together with data about the web resources results in a document set that matches the query. Some search processes might use metadata to drive or enhance the search. What MIRADOR aims to provide is an ability to generate tailored descriptions of the document set identified in the search process, using metadata and using user profiles for the tailoring.

2. Background

There have been a number of recent developments in providing web resources that have contributed to our approach.

The concept of metadata is increasingly gaining currency as a mechanism for enhancing the usefulness of internet resources, and there have been a number of initiatives around formalising and standardising frameworks and architectures for its use. The Dublin Core is one example of such an initiative, an attempt to identify a metadata set for describing electronic resources, and there have been others. Thiele[1] provides a good overview of the literature around the Dublin Core Workshop series.

In the wake of these initiatives, there has been an increase in the number of web resources that include metadata. This is not without its own problems, however. We have noticed for example that the inclusion of metadata in web resources is often patchy, and occasionally we might find a document whose metadata bears little relation to the resource it describes. Like comments in a program, because the metadata is not as visible as the resource itself, it can often be forgotten when a resource is changed or upgraded. In the most extreme case an author might use a metadata template, perhaps copied from another resource, and not ever get round to completing or changing the metatdata for the new resource.

There has been progress towards metadata standards for describing educational resources. In particular, the IMS Project (Instructional Management Systems), a cooperative of academic, commercial and government organizations, looking at internet architecture for learning, is investing some energy in metadata, and together with ARIADNE (Alliance of Remote Instructional Authoring and Distribution), has come up with a schema for education metadata.

Briefly, the ARIADNE proposal describes a schema of information, in 6 mandatory categories. These (currently) are:

- general information on the resource itself
- semantics of the resource
- pedagogical attributes
- technical characteristics
- conditions for use
- meta-metadata information

Within these, various descriptors are proposed, not all of them mandatory. There is

currently a total of over about 25 descriptors, and within these, the IMS proposal describes over 80 elements, in a hierarchy.

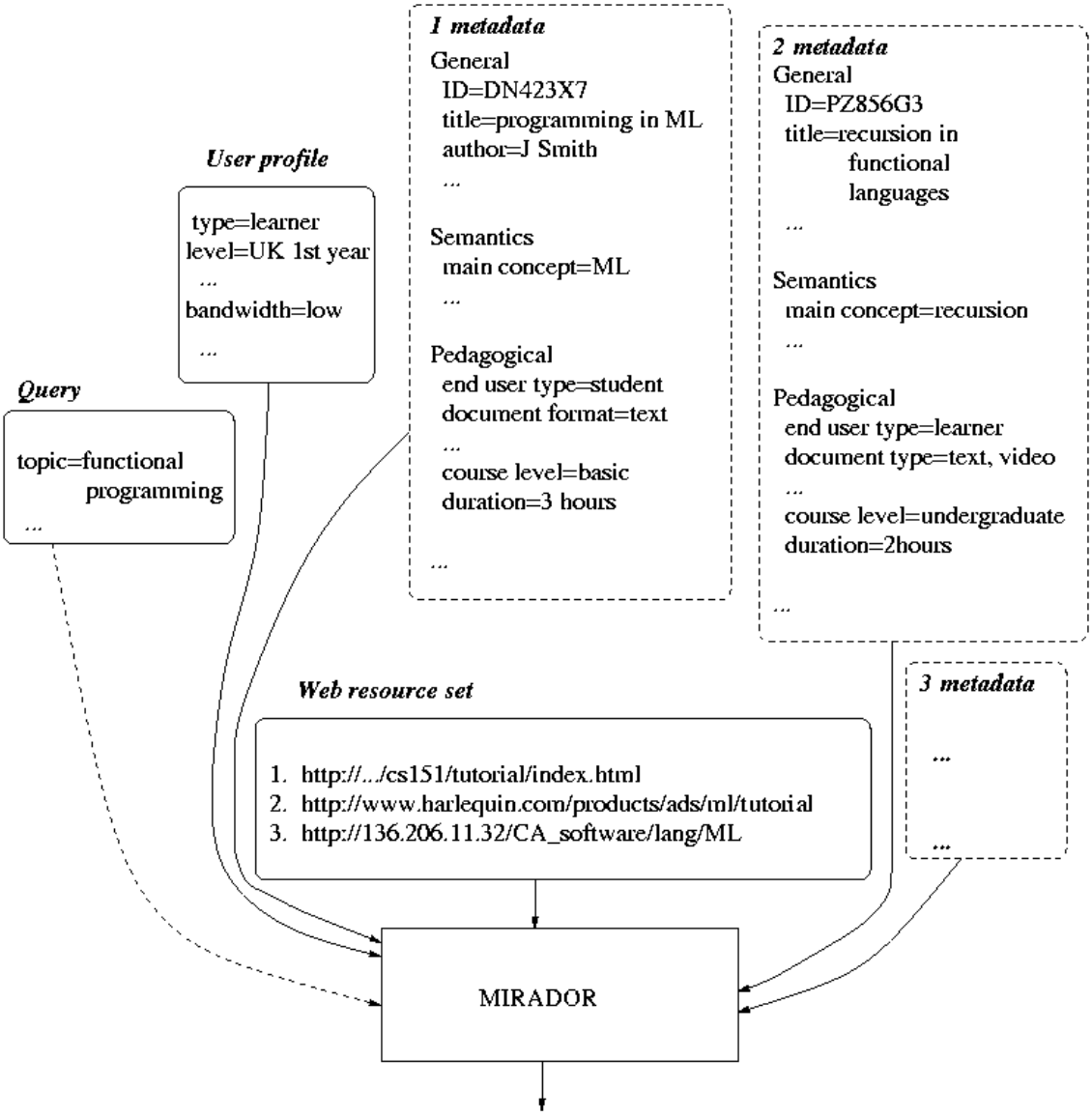


Figure 2 An example of the kind of output we aim for MIRADOR to produce

In the computational linguistics arena, natural language generation techniques are being used to generate summaries of documents, for information retrieval [2]. In contrast to this,

our approach is to use metadata as the underlying knowledge base, and apply natural language generation techniques in generating tailored descriptions of resources. One important advantage that metadata gives over using document content as the source data, is that metadata is available for non-text resources, such as those containing video, audio, or graphics. The approach taken by Amitay[3] recognises these limitations with non-text documents. Instead of using the resource of interest itself, she looks for descriptions found in textual resources that point to the resource of interest, and aims to determine which such description is the most appropriate. Our approach is instead to use the metadata found with the resource, to generate a coherent description of a number of resources, that is tailored to the context of use.

So, combining relevant aspects of these recent developments, we propose to address the need for better mechanisms for finding web resources, by generating natural language descriptions of existing multimedia resources from metadata about the resources, and tailoring these descriptions to take into account the educational context and user need. We have constructed the example in Figure 2 to demonstrate this, using categories based on the ARIADNE-IMS educational metadata recommendations.

3. A preliminary investigation

As a prelude to establishing what kinds of tailored descriptions would be useful to users of web resources, we have done a study of some existing descriptions. Our aims in this study are to:

1. determine the structure of human authored document descriptions, which will inform the design of the descriptions we might aim to generate
2. establish how the content of some human-authored descriptions matches the information in a recognised metadata schema (in particular, the ARIADNE-IMS master schema, which has been proposed for educational resources).

EEVL, the Edinburgh Engineering Virtual Library, maintains a searchable catalogue of reviews and links to engineering-related web sites. Described by Moffat [4], it was established as a project under the Electronic Libraries Programme. The EEVL database contains descriptions of nearly 4000 engineering-related web sites, and we have taken a sample of 22 of these (descriptions of tutorials, about computing topics) for our analysis. These are descriptions of single documents, and our aim is ultimately to provide descriptions of multiple resources, including some contrast and comparison. However, the single document descriptions are a useful starting point for identifying the kinds of information such descriptions contain.

The resources being described are generally text documents, but some of them include video clips, audio, images, etc. Two example descriptions are shown in Figure 3. We have identified (so far) some 20 different information types, which together cover the content of the descriptions. These include things like *Written_for*, *Written_by*, *Written_why*, *Keywords*, *Aimed_at*, *Time_to_complete*, etc. Any one description may have text that pertains to any of these 20 information types.

The IMS metadata master schema is based on the IEEE LOM V2.2 Working document, jointly authored by IMS and ARIADNE. Some, but not all, of our information types have an obvious mapping to this schema. The IMS schema, in contrast to the Dublin Core, has a hierarchy of sets of metadata, containing a total in excess of 80 fields.

```
6. PLC Tutor

The PLC Tutor provides a complete non-vendor-specific guide to
programmable logic controllers. It offers an introduction, and
sections on basic programming, advanced programming, wiring, and
links to manufacturers. New chapters are introduced to the
```

tutorial on a regular basis. The site can also be downloaded, if required.

13. Transmath - a CBL mathematics tutor

The Transmaths project aims to address some of the problems experienced by increasing numbers of first year undergraduate scientists and engineers, who arrive at University with inadequate mathematical knowledge and skills, by providing them with a self-paced, user friendly computerised mathematics tutor. The Mathematics departments of Imperial College and the University of Leeds were awarded a grant under the Teaching and Learning Technology Programme (TLTP) to produce CBL material for the remedial teaching of mathematics.

The Transmath Web page provides further information about the project, a list of available modules, an ftp server for downloading Transmath, articles written by the Transmath team, and an evaluation of TMP and Transmath software. Other sites of interest are also available.

Figure 3 Example descriptions, taken from the EEVL database

Initial findings

Most of the information we have in the descriptions can be directly connected with fields within the ARIADNE categories General, Technical, Pedagogical, and Semantics. The areas where this is not the case are minor:

We have noticed that many descriptions will themselves have hyperlinks to resources other than the one they are describing. This is information that is not easy to cater for in the metadata schema. The closest we can find is the information in the Relation category, which is described as 'characteristics of the resource in relationship to other resources', but this is more general than we would like.

In the IMS schema, keywords are a sub-level within the Semantics category, pertaining to the Concept or the Discipline. This does not always fit with the use of keywords in our sample descriptions, which might sometimes pertain to the educational goal, or the nature of the resource, for example.

Many of the information categories will have contributions from more than one language fragment, while others will have only one. For example, there may be a number of educational concepts mentioned in the description; there may be a number of prerequisites for learners wishing to use the resource. Though usually such collections of information are close to each other in the description text, there is no requirement that this is the case. In particular, hyperlinks to other resources may be distributed through various parts of the description.

We also note that an important property of text descriptions is that they provide a coherent organisation of information, that itself conveys some semantic content. An analysis of this organisation takes us beyond the flat representation that is obtainable from metadata. In particular, the Dublin Core metatdata has no hierarchical structure at all; the ARIADNE-IMS schema has a hierarchy which is inflexible. This suggests that in generating descriptions we must consider this indirect semantic content and underlying communicative goals.

Natural language prose is felt to be more useful than say just listing the metadata information for a number of reasons. It is possible to focus the information in a way that is tailored to the user's needs and it allows emphasis and stress of certain information. In addition, a text description is an appropriate way in which to provide the kind of comparison we envisage. A small study is planned, to verify this, comparing the usefulness of text versus tables in this context.

4. Concluding remarks

The MIRADOR project is one that aims to bring together recent developments in metadata, educational resources, and natural language generation, in a novel way, which will provide some help to users of networked educational resources, in dealing with the volume of resources that are available.

We aim to develop a system to provide better descriptions of resources, and take as our starting point an analysis of professional human authored descriptions. This analysis suggests that the metadata we plan to use as our source data is, if complete, a mostly adequate resource, but we have identified some limitations. Our analysis also suggests ways of structuring descriptions.

We now plan to work on implementation issues and move on to consider how descriptions can be tailored to user and query.

Bibliography

1. Thiele, Harold, *The Dublin Core and Warwick Framework* D-Lib Magazine, January 1998, ISSN 1082-9873. Available: http://www.dlib.org/dlib/january98/01thiele.html [Accessed February 1999].
2. McKeown, Kathleen R., Jordan, Desmond A. and Hatzivassiloglou, Vasileios *Generating Patient Specific Summaries of Online Literature.* AAAI Spring Symposium on Intelligent Text Summarisation, pp34-43, Stanford, 1998.
3. Amitay, Einat. (1998). Using common hypertext links to identify the best phrasal description of target web documents. In *Proceedings of the SIGIR'98 Post-Conference Workshop on Hypertext Information Retrieval for the Web*, 1998. Available http://www.mri.mq.edu.au/~einat/sigir/ [Accessed April 1999].
4. Moffat, Malcolm, *An EEVL solution to engineering information on the Internet* Aslib Electronics Group 38th Annual Conference, 15-17 May 1996. Available: http://www.eevl.ac.uk/paper1.html [Accessed February 1999].

Conceptual Documents and Hypertext Documents are two Different Forms of Virtual Document

Sylvie Ranwez and Michel Crampes

Laboratoire de Génie Informatique et d'Ingénierie de Production
EMA - EERIE, Parc Scientifique Georges Besse
F-30035 Nîmes Cedex 1
ranwezs@site-eerie.ema.fr, mcrampes@ema.fr

Abstract

When posing simple queries on the Internet, users often find themselves facing relatively incoherent and poorly organized groups of items of information. They should be helped in their navigation through them. This help can take the form of Conceptual Documents that are adapted to their particular state and preferences.

In this paper we give a definition of ***Virtual Documents*** *and we study two particular cases:* ***Hypertext Documents*** *and* ***Conceptual Documents****. We show how Conceptual Documents can improve data retrieval and quality of knowledge transfer. We propose an approach for generating conceptual documents and we present an application based on this approach.*

Keywords*: Virtual Documents, Conceptual Documents, Hypertext, Conceptual Navigation.*

1. Introduction

Searching for information on the Internet, users find themselves faced with collections of pieces of information which they bring to light either by browsing or by using search engines. These collections often have little coherence and users have to filter and organize them in order to make them exploitable. When they do this, they synthesize a form of document suited to their needs. In what follows, we call the individual pages or pieces of pages which users unearth 'Information Bricks' (IBs). We call a Real Document (RD) the composite document which they eventually synthesize out of these bricks.

Our research focuses on techniques that would allow the production of these real documents from raw information as coherently as possible. A collection of information bricks with techniques suited to building a real document is called a Virtual Document (VD) - one which is not itself a real document but which contains the specifications necessary for producing one.

This paper analyses first what characterizes coherency in a document, then how it is possible to produce a VD from a set of IBs. Two forms of VD are analyzed: Hypertextual and Conceptual. A Real Document is presented as the result of a VD and specific circumstances. The circumstances differ according to the type of VD - whether it is hypertextual or conceptual. Hypertext documents come into being through the user's own browsing through hyperlinks. Conceptual documents through a conceptual engine and the user's specifications.

Finally we give a presentation of a project under development in our laboratory for designing a foundation for building conceptual documents.

2. Preliminary definitions

A **document** (from the *Latin documentum*) means "a thing used for giving instruction". Often it is written and can be used as proof, information or testimony.

A document is a tool used for transferring knowledge. Therefore it must be structured in a way that optimizes the user's comprehension. This structure might be as follows:

- A document is composed of **different parts** that we call bricks.
- Each part has a **style** and plays a particular **role** in the knowledge transfer. The role depends on the place of the part in the document.
- Each part has a certain **volume** or **size.**
- Between bricks there are **transitions** that express the **chronological** or **causal** links that join them together.
- In a given document, the vocabulary comes from a single **domain ontology** and, consequently, the same word should always mean the same thing.
- Finally the document has an **entropy value** which represents its degree of complexity according to Shannon's theory of information.

This document is called a **Real Document** (RD) since it can be consulted without any change, i.e. in its present state.

The thing we call an **Information Brick** (IB) is a fragment of a document, rendered on one (at least) medium, characterized by a conceptual model and insertable into a real document. Once a set of IBs exists, the building of a real document takes the form of selecting the pertinent ones then organizing and assembling them. Bricks can be nested since they can be segmented into sub-bricks. The size of an IB depends on the author's wish (how deep he explains a concept), and on the content itself: a pen description will probably be shorter than a plane description.

The document from which IBs are extracted is called a **Source Document.**

The most common definition for **Virtual** is: being in essence or effect but not in fact; in other words, being in a state of possibility. In the following we will adhere to this definition; it is one widely used in data processing - in for example the term 'virtual memory': something which appears functionally for a given user without taking into account the physical structure or the logic used.

In accordance with the two definitions given above, we can define what a Virtual Document is.

3. Virtual Document: a definition

A **Virtual Document** (VD) is a non-organized collection of Information Bricks (IBs) associated with tools and techniques allowing the creation of a Real Document (RD).

By analogy with object-oriented languages, a Virtual Document is a composite class, IBs are components, and techniques and tools are construction methods. A Real Document can be seen as an instance of this class.

A more formal definition for a Virtual Document might be:

VD = { IB } + Methods allowing the generation of a finished IB sequence.

The methods must take into account:

- predefined links between IBs if any;
- any parameters supplied;
- user actions;
- a specific strategy for methods.

A Real Document is then a sequence of IBs generated by the methods in concordance with the user's specification.

4. Different forms of Virtual Document and their characteristics

In this part we will differentiate two kinds of Virtual Document: Hypertextual and Conceptual.

4.1 Hypertext Documents (HD)

By analogy with the vocabulary used in object languages, a **Hypertext Document** is a sub-class of a VD.

An HD is composed of hypertext information bricks that have predefined connections. The links contained in these bricks (hyperlinks) lead to other bricks - the whole constituting a graph. The method which allows the building of a Real Document out of an HD (the "constructor" of an object-oriented language) is the user's browsing through the document - i.e. the visit paid to this graph.

A hypertext document is by definition a VD because its final form depends entirely on the wishes of the user. The route taken in visiting the document is known only by the user; it is not preset. A particular case of a HD is the one where there is only one link at the end of each page - and it leads to the following page. In this case the HD is also the RD.

The formal characteristics of a VD are:

HD = {Hypertext IB} + User's browsing.

- The predefined links are the hypertext links that are in the IBs;
- There are no given parameters;
- User actions are limited to clicks on hyperlinks;
- The strategy is specific but implicit in each user's mind.

4.2 Conceptual Documents (CD)

A **Conceptual Document** is a Virtual Document from which it is possible to build a Real Document dynamically and at any moment the user asks for one. Contrary to HDs, the bricks that compose a CD can have several formats. These bricks are selected via the semantics of their contents. They can be accessed via the Internet or locally (CDROM, DVD, hard disk,...). In this case, the methods used to create a Real Document consist of an engine and the user's specifications. The engine is in charge of selecting the IBs, organizing and assembling them, but it does this in obedience to specifications defined by the user.

Among these specifications can figure economic constraints such as reading time. This point is significant since it constitutes one of the major differences between HDs and CDs. Indeed, we can have HDs whose links take into account the semantics of the IBs referred to, but time will never play a role in the browsing. However it is one of the things users keep firmly in mind.

With our formalism, we can write:

CD = {IB}+ Engine and specifications.

- Any and all links are possible between bricks since the links will be the result of a conceptual evocation process [CRA 97][CRA 98]. The set of bricks constitutes a special graph called a "clique".
- The parameters are given by the user (time, final document size, data concerning their requirements, conceptual focus, etc.) ;
- The user's actions amount to selecting the required parameters;
- The engine can be directed to follow a precise strategy. This is important in particular in pedagogical applications.

4.3 Other characteristics for VDs

It is possible to distinguish homogeneous VDs from heterogeneous VDs.

4.3.1 Homogeneous Virtual Documents

A VD is **homogeneous** if all its bricks come from the same source document. Thus they all have the same author or group of authors.

A homogeneous document has the following properties:

- IB **size** is nearly the same for each brick. There is no great difference in size because the same method is used for partitioning all the bricks of any one document;
- Brick **style** is the same for all. Because all the bricks come from the same author and from the same source document, scriptural and graphical styles do not change;
- It is possible to give indications to guide **dictum order**. The source document can contain causal links and information that can serve during brick ordering;
- The **role** of each brick is defined in relation to the dictum order;
- **Text continuity,** and the use of linking words, is related to the framework of the source document;
- **Entropy** is high: there is little redundancy,
- There are no contradictions between the different bricks (the author may have respected some basic guidelines yielding coherence, readability and restriction to suitable vocabulary);
- Finally, the **ontologic domain** is unique: words always have the same meanings (not unrelated to the remark above about the quality of a document).

4.3.2 Heterogeneous Virtual Documents

Heterogeneous Virtual Documents are documents composed of bricks that can have several different origins – Internet, CD-ROM, etc. – and thus several authors. Their characteristics are as follows:

- The **size** of the different bricks making up a document can differ.
- Their **styles** can vary widely - to the extent of shocking the user when assembled together into a composite document.
- There is no **partial order** between IBs (causal, presence etc. links are not defined).
- The **role** of each brick is unspecified.
- There are no transitions between bricks.
- **Entropy** may be low: redundancy may occur, with several bricks treating the same subject.
- There may be contradiction - between authors or points of view.
- The **ontologic base** is heterogeneous and a word may have different meanings in different bricks.

In such documents, problems of coherence are likely. Transitions between IBs will need particular treatment.

5. How to create Real Documents from Conceptual Ones?

The constructing of an RD has three phases to it. The first one is Information Retrieval (IR), and it takes into account size constraints. The second one is the building of the real document; this includes the filtering of the retrieved IBs, the ordering and assembling of these IBs. The final one consists in displaying the document through a convenient interface (HTML for example).

To do this we need an engine. We have constructed what we call a Conceptual Evocative Engine (CEE), one which uses the semantics of the IBs to do the selecting and building of the final document [CRA 97] [CRA 98].

To allow reusability of IBs, they have to be qualified with metadata. Qualification needs a description method. We have developed such a method through a Document Type Definition (DTD) and the corresponding language written in XML. Once the qualification is done the CEE can identify each brick by the semantics of its content and build up conceptual links between those bricks.

Brick qualification using a DTD and associated treatment has several advantages, amongst them the apprehension of constraints of different sorts - narrative and economic for example.

First, narrative constraints:

- Using the semantics of the IBs' content, information retrieval is improved and it is possible to avoid contradictions.
- If the DTD is based on a unique ontology, problems of synonymy and homonymy are avoided.
- It is possible to express constraints such as causality in the description of a brick that will be used during the ordering of the IBs.

Second, economic constraints:

- Time constraints can be respected, using the time parameters included in the description. During the selection of IBs, the CEE can eliminate any bricks that are too long, selecting only those that correspond to the time available to the user.
- It is possible to confine selection to bricks of the same size, improving thus the homogeneity of the final document.

When the engine has selected IBs which meet the user's needs, it has to assemble them. This is achieved by taking into account not just user preferences and narrative and economic constraints but also information such as the educational curriculum the user has followed or is following. The result is a real document adapted to the specific needs of the specific user.

6. Application: The Karina project and the associated DTD

The theoretical considerations set out above constitute the basis of an application - that we call Karina - that we have developed in our laboratory.

Karina is a project jointly undertaken by the Alès School of Mines (l'Ecole des Mines d'Alès or EMA) and the Marseilles Higher Engineering School (l'Ecole Supérieure des Ingénieurs de Marseille or ESIM) within the framework of the French Ministry of Industry's call for projects for the Information Highway. This particular project aims to furnish teachers with tools enabling them to use the Internet for making courses available and also for making use of course bricks provided either by other teachers or by alternative sources - electronic newspapers etc. Initially our application is oriented towards distance learning, but we wish it to remain general enough to embrace cultural and leisure activities as well.

The conceptual evocation engine has served as the object of models [CRA 98] for realizing narrative abstracts in the domain of interactive television. It is now subject to further development, with the aim of finding application in the domain of teaching [RAN 98].

7. State of the art

In adaptive hypermedia systems, the aim is to find a compromise between guiding users and letting them browse on their own [BRU 96] [GRE 97] [STE 97] [WEB 97]. This work concerns essentially the adaptation of user-browsing to an already established hypergraph [BRU 98]. It does not aim at the construction of new links and their organization in response to user needs. The approaches cited above are attempting to find ways of adapting pre-existent hypermedia, not dynamically constructing routings through ad hoc collections of bricks borrowed from other documents.

The sharing of resources though makes document indexing necessary. Document description articulates around complete documents and makes use of either specific descriptors [MAR 97][BAR 98] or descriptors already established as standards or recommendations [DUB 97][MARC]. This latter category includes the IMS (Instructional Management System) recommendation for educational documents [IMS 98]. The aim of the project is to furnish specifications for the description of educational materials using a system of meta-data.

The XML (eXtensible Mark-up Language) language allows the description of electronic documents by means of a DTD (or Document Type Definition). The use of DTDs for

internet documents is recent yet already well-established because, for example, a preliminary version of the MARC standard DTD is available [MARCDTD]. XML is intended to be evolutive since it allows for the fusing of several DTDs - not unlike the principles of inheritance in the object world [W3C 98].

Other forms of document description are visited: [BAL 97] for instance proposes an object architecture for modeling electronic documents. In Karina, we use a similar reification technique to transform a descriptor into an XML element without however situating our model in the pure object world.

The authors of [BRA 98] use HTML comments to annotate documents in order to be able to implement adaptive changes. In the Karina project, the entities forming the basis of conceptual browsing are explicitly declared as XML elements - not hidden in comments.

[GRE 98] proposes a technique for automatically generating links between documents treating any one subject. This permits grouping the responses to a search on that subject. In order to extract a content semantics, the approach is to note the words and their frequency of use in a document - including closely related words (this via relations which can be established by WordNet links). The main problem here is in speed of execution.

Starting from the linear form of the conceptual graph, we have explored the use of light versions of weighted conceptual graphs for effecting conceptual evocations between documents [CRA 97] [CRA 98].

8. Conclusion

In their browsing and searching on the internet, users need guidance. They need a system capable of creating documents adapted to their precise requirements and demands; documents which come into being as a result of the application of certain circumstances to a virtual document.

After defining these different types of document, we have concentrated on one of them - the conceptual one. We put forward a system permitting the creation of such documents and the production of real documents adapted to the needs of their users.

Our approach focuses on modes of conceptual browsing. These operate on collections of information bricks which are qualified - that is to say containing inside themselves meta-information related to their content and to any constraints limiting their use. A conceptual evocation engine (CEE) has been modeled and is currently the object of further development.

The process of constructing real documents out of their conceptual counterparts requires the definition of rules of narrative construction and constraint optimization algorithms - this because a real document has to obey constraints of size and time. Our theoretical work now bears on this aspect of the problem. We are attempting to establish the formal bases of a language for conceptual document description - and thus also for virtual document construction.

References

[BAL 97] Baldonado M., Chen-Chuan K.C., Gravano L.? *Metadata for Digital Libraries: Architecture and Design Rationale.* Actes. DL'97 ACM Digital Library '97*?* Philadelphia., PA., USA, July 1997, ACM Press, pp. 247-253.

[BAR 98] Barthélémi S., Loubier M., Pinon J.M., *SEMUSDI, SErveur MUlitimédia pour les Sciences De l'Ingénieur*. Actes du congrès NTICF'98, INSA de Rouen, 18-19-20 Novemvre 1998.

[BRA 98] De Bra P., Calvi L., *2L670: A Flexible Adaptive Hypertext Courseware System*. Actes HyperText'98, Pittsburgh., PA., USA, June 1998, ACM Press, pp. 283-284.

[BRU 96] Brusilovsky P., Schwartz E., Weber G., *A Tool for Developing Hypermedia-Based ITS on WWW*, Position Paper for ITS'96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, Montreal, June 10th 1996.

[BRU 98] Brusilovsky P., *Methods and Techniques of Adaptive Hypermedia. Adaptive Hypertext and Hypermedia*, Brusilovsky, P., Kobsa, A., et Vassileva J. eds. Kluwer Academic Publishers, 1998.

[CRA 95] Crampes M. Composition Multimédia dans un contexte Narratif. Modèles et Maquetage Basé sur une Architecture Agents. PhD Thesis, University of Montpellier II, 1995.

[CRA 97] Crampes M. Auto-Adaptative Illustration through Conceptual Evocation in Proc. DL'97 ACM Digital Library '97 (Philadelphia., PA., USA, July 1997), ACM Press, pp. 247-253.

[CRA 98] Crampes M., Veuillez J.P., Ranwez S., *Adaptive Narrative Abstraction* Actes. HyperText 98, Pittsburgh., PA., USA, June 1998, ACM Press, pp. 97-105.

[DUB 97] *Dublin Core Metadata Element Set: Reference Description*, http://purl.oclc.org/dc/about/element_set.htm, 1997.

[GRE 97] Greer J.E., Philip T. *Guided Navigation Through Hyperspace*, Actes Workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society, Kobe, Japan, 18-22 August 1997.

[GRE 98] Green S.J. *Automated Link Generation: can we do better than term repetition?* Seventh International World Wide Web Conference, Brisbane, Australia, 14-18 April 1998.

[IMS 98] Educause, *Instructional Management Systems.* http://www.imsproject.org/metada, 1998

[MAR 97] Marchionini G., Nolet V., Williams H., Ding W., Beale Jr. J., Rose A., Gordon A., Enomoto E., Harbinson L., *Content + Connectivity => Community: Digital Resources for a Learning Community*. Actes Second ACM Digital Library conference, Philadelphia, PA, USA, July 1997.

[MARC] Library of Congress; Network Development and MARC Standards Office. *MARC STANDARDS Machine-Readable Cataloging,* http://LCWEB.loc.gov/marc/.

[MARCDTD] Library of Congress; Network Development and MARC Standards Office. *MARCDTD,*http://lcweb.loc.gov/marcdtd/mrcbfile.dtd.

[RAN 98] Ranwez S., *Formalisation d'Ontologie Pédagogique (incluant matériel et procédés didactiques) et raisonnement sur cette ontologie pour l'élaboration de cours adaptatifs, suivant différentes Stratégies Pédagogiques, dans un système de formation continue disponible via Internet*. Rapport interne LGI2P, Ecole des Mines d'Ales, 1998.

[STE 97] Sterb M.K. *The difficulties in Web-Based Tutoring, and Some Possible Solutions*, Actes Workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society, Kobe, Japan, 18-22 August 1997.

[W3C 98] W3C, *Document Object Model (DOM) Level 1 Specification Version 1.0* W3C REC-DOM-Level-1-19981001, http://www.w3.org/TR/REC-DOM-Level-1/, 1 October 1998.

[WEB 97] Weber G., Specht M. *User Modeling and Adaptive Navigation Support in WWW-based Tutoring Systems*, Actes UM-97, Cagliari, Italy, June 2-5, 1997.

When Virtual Documents Meet the Real World

Stephen J. Green[1], Maria Milosavljevic[2], Robert Dale[1] and Cecile Paris[2]

[1] Language Technology Group
Microsoft Research Institute
Macquarie University
North Ryde NSW 2113
Australia
{sjgreen,rdale}@mri.mq.edu.au

[2] Intelligent Interactive Technologies Group
CSIRO Mathematical and Information Sciences
Locked Bag 17
Sydney NSW 1670
Australia
{Cecile.Paris,Maria.Milosavljevic}@cmis.csiro.au

1. Introduction

We have been involved in an ongoing project involving the creation of virtual documents using techniques drawn from the area of Natural Language Generation (NLG; see Reiter and Dale (1999) for an up-to-date overview of this field). In the past year, we have taken this work to a new level, by attempting to scale up the techniques we originally developed on small hand-constructed knowledge bases (KBs) to a stage where they cam be applied to a large KB that was automatically built from a database of museum objects.

This experience has illustrated some important principles to keep in mind when virtual documents on the Web meet the difficulties and inadequacies inherent in real-world data. In addition, we have gained some insight into how generation techniques can be used to produce virtual documents in multiple languages.

2. Virtual documents and NLG

The forerunner of most of the systems that we have built is the Peba-II system (Milosavljevic et al., 1996), which automatically generates descriptions and comparisons of animals. These descriptions and comparisons are based on a model of the user, so that they can include references to animals that the user has already seen described.

As an experiment in seeing how domain-dependent these techniques were, we took the Peba-II system as a base and built a new database containing the kinds of objects that one would expect to see in a museum of computing. This system, Power, could be used to produce exactly the same kinds of descriptions and comparisons as Peba-II, with only a new KB and the addition of some new lexical items.

The problem with the approach taken in these two systems was that the KBs had to be constructed by hand, a time-consuming and difficult task at best. Because of this, the KBs were small, although of quite high quality. Despite this, the systems have been quite successful and have shown the feasibility of using NLG techniques in a near real-time environment like the Web (see Dale et al. (1998) for more information.)

3. Automatically acquiring a knowledge base

Our success in porting Peba-II to a new domain lead us to consider how we could automatically acquire a much larger KB of objects to describe. Our interest in the museum domain is not unique - The Intelligent Labelling Explorer (Hitzman et al., 1997) project has also focussed on generating descriptions of museum gallery objects. The Powerhouse Museum was willing to provide access to the database of their Collection Information

System (CIS). This database consists of approximately 300,000 objects, most of which are in storage at the museum's various facilities because of a lack of floor space in the museum's exhibition halls.

```
<rec num=12798 id="H4448-513">
OID:  H4448-513
INT:  Part
LOC:  TH2.STEP.6A
LOD:  27/11/1997
OBN:  Boots
OBS:  Balmoral boots, elastic sided, pair, women's,
      patent/kid/leather/elasticised fabric/wood,/brass prize work,
      [Gundry & Sons], England, c.1851; 1862-1869.
DES:  Balmoral boots, elastic sided, pair, women's,
      patent/kid/leather/ elasticised fabric / wood /brass, prize
      work, [Gundry & Sons], England, c.1851; 1862-1869. Pair of
      women's elastic sided boots (Balmoral), with wooden filler, of
      welted construction with rounded toes featuring peaked caps and
      stacked heels. The uppersconsist of a patent golosh, seamed at
      the back, glace kid leg, seamed at front and back, and elastic
      sides extending to the golosh. The uppers are decorated with
      oval stitching at the edge of caps and scallops at the throat
      of golosh. The leather heel is fine wheeled, featuring a top
      piece with brass nailed edge. The black leather sole features a
      sueded forepart with brass nails, as well as an internal clump
      and brass hinged section for extra strength and a brown
      polished ridged waist with black edge. Reputed to have been
      made by Gundry & Sons. (See object file for specialist report
      by June Swann)
MDE:  Gundry & Sons; London, England
MDN:  1965 list says "made by Gundry & Sons, Soho Square." Swann says
      hinged device to increase flexibility is unusual. Similar
      screws on H4448-515. Note hinged sole in 1862 exhibition. She
      finds no information about Box in information she has about the
      1851 exhibition, though William Walsh is mentioned in
      connection with a pair of shoes. Patent 558, 5 March 1861,
      granted to J.M. Carter, a similar sole with 2 cuts across the
      tread and 4 rows of screws "for soldiers, riflemen, sportsmen.
      The inner sole is whole and contains pitch." It is not possible
      to confirm whether these boots contain pitch.
DAT:  c 1851 - 1869
MAR:  Interior obscured by last, no marks on exterior
DIM:  Length 248 mm Height 31 mm Overall Height 160 mm Width 58 mm
</rec>
```

Figure 1: A well-populated database record

We selected as a subset of the items in the database only those objects on display in the museum. We also included in this set all those objects in the database that were a part of something on display, even if these particular objects were not on display. This subset amounted to approximately 15,000 objects. Figures 1 and 2 demonstrate the range of data quality in the database. The entry in Figure 1 provides a rather complete description of the Balmoral boots, including a discussion on the construction and history of the boots. The entry in figure 2, on the other hand, provides no information beyond that required for a database record to actually exist! It is worth noting that this database was intended for internal use by curators only, and not intended to be used to generate object descriptions

for public consumption.

```
<rec num=15463 id="PROP/EXP/82">
OID:  PROP/EXP/82
INT:  PROP
LOC:  EH2.EXP.3H
LOD:  16/02/1996
OBS:  Aerosol deodorants/colognes (2), "Lynx" & "Australis".
DES:  Aerosol deodorants/colognes (2), "Lynx" & "Australis".
</rec>
```

Figure 2: A sparsely populated database record

In addition to the database, we were provided with a hierarchical thesaurus of objects that had been prepared by the museum. With these two information sources, and with lists of countries and materials extracted from the Macquarie Thesaurus we could begin the process of extracting a KB from the database. The extraction process is broken down into five basic steps:

1. Normalisation of the database: This is to ensure that each record is surrounded by an SGML-style rec tag, and that each field of an entry is on a single line. This eases the processing requirements at the further stages.
2. Extraction of dimensions: In this step, a Perl script extracts the dimensions of the objects. This information resides in easily identifiable fields (e.g., the DIM field in Figure 1) and the information in that field is structured and can be decomposed into its subfields (e.g., length and height).
3. Extraction of thesaurus categories: This step involves trying to identify the thesaurus category that applies to each of the objects in the database. This is normally found in the OBN (Object Name) field and corresponds to an entry in the Powerhouse's thesaurus.
4. Extraction of names, materials, makers, locations, and dates of construction: This involves extracting information from the textual information contained in the database records. Most of our work here so far has focussed on the OBS (Object Statement) field. This field is supposed to include information encoded in a standardised and rigorous way. However, in practice, not all the information that is supposed to be included is present, or it is present in a different order, or format, from the norm. Yet, with the help of information from the Powerhouse's thesaurus and the lists of materials and countries, we were able to identify information such as date of manufacturing or purchasing, materials and location.
5. Extraction of PART-OF and A-KIND-OF information: We used the OID (Object ID) field to determine the PART-OF hierarchy for the database. For example, in the database record shown in Figure 1, the OID H4448-513 indicates that this object is the 513th part of the object with OID H4448 (in this case the Balmoral boots are part of a large collection of footwear). According to the database specifications, an object may have parts, sub-parts, and sub-sub-parts.

Clearly the quality of the KB entries created will depend greatly on the database they are extracted from. For example, of the 15,483 records that we received in the database dump, only 9,887 (in other words, around 64% of the total) actually have an OBN field. Furthermore, of the 9,887 objects that have OBN fields, only 7,751 are valid object names, that is to say that the Object Name assigned to the object appears in the museum thesaurus. Thus, about 50% of the database entries do not provide any information about the types of the objects.

Even a high-quality entry, such as that in Figure 1, is problematic, as much of the information that we would like to put in a description is "locked-up" in text. In Figure 1, we see this in the description of the construction of the boot. Although much recent work in Natural Language Processing is oriented towards extracting useful information from texts

that have some similar characteristics, current techniques are simply not up to the task of performing this extracting reliably in such a way that results are reusable for our purposes. We cannot simply use the text directly, as we cannot guarantee that the text will be well structured or even relevant. In addition, the text may contain information that the public viewing these descriptions are not meant to know. A more-structured database would make this task significantly easier. Figure 3 shows the generated text for the database entry in Figure 1.

Figure 3: A description of the Balmoral boots produced by the PowerTNG system

4. Navigating through a virtual document space

In the Peba-II and Power systems, the task of navigating the virtual document space was accomplished by selecting items of interest from a list and then following hypertext links around the object hierarchy. This is sufficient when there is only a small number of objects and a user can easily look through a list of them. When we move to a KB consisting of thousands of objects, however, we need to rethink our navigation strategy. It is, to say the least, infeasible (and perhaps impossible in current browsers) to have a user select an item of interest from a list of 15,000.

The obvious choice for a navigation strategy is to use the object hierarchy provided by the Powerhouse. This has the advantage that we can use a structure that was built by humans and which therefore should represent the actual relationships between the objects in the database. Given the thesaurus and the generated KB, we can produce a hierarchy that will allow a user to navigate from the top level to a specific object. Currently, this navigation hierarchy is a static set of pages that is generated when the KB is regenerated, but there is no reason why it cannot be dynamic too.

As the location of the objects in the museum is also encoded in the database entries, we built a navigation hierarchy based on location. The location information includes the exhibit that an object is part of as well as the case in which it appears. We found, however, that such a hierarchy offered little help to users, since they had no idea what was in the

exhibits, let alone which objects were gathered in something called "Case 6A".

While using the thesaurus-based navigation hierarchy allows a user to navigate to an object by traversing through classes and subclasses, it will not allow them to navigate directly from one object to another. Our aim in building the generation component was to make sure that any systematic relationships between objects (e.g., that one is a part of another, or they were manufactured by the same artisan) is expressed in the virtual document as a hypertext link. This is implemented simply as a set of inverted files to which the generation component has access.

Our initial approach was to inline all of these links into the text, but this led to some confusion about the discourse intention of following the links (see Milosavljevic and Oberlander (1998) for a discussion of dynamic hypertext as discourse). In the original Peba and Power systems, only the names of objects were linked. The discourse intention of following one of these links was clear: "describe this animal" or "describe this computer". However, when more than one kind of entity is linked, the intention is less clear. For example, if the user follows a link whose anchor is the word leather, will he or she be taken to a description of the material leather or to a listing of the objects in the database that are made of leather?

Our solution was simply to make the discourse intentions for links other than links to named entities very explicit, leading to constructions such as See other objects made of leather appearing at the end of the descriptions of objects.

5. Changes in functionality

Although we are claiming to be porting the Peba-II/Power system to an automatically generated KB, in fact, we have completely re-written the system to take advantage of some of the lessons learned during the development of the previous systems. As with any re-write, there have been some changes in functionality.

5.1 Comparisons

Unlike the ancestor systems, the current system will not generate comparisons. There is no fundamental reason why this cannot be done, but there is no obvious way to introduce a comparison to the user. In the original systems, comparisons were generated either by selecting two objects from two lists of all the objects known to the system or by selecting another object when looking at a description of some object. Needless to say, this is ineffective when there are 15,000 objects to choose from.

Another factor limiting the scope for providing comparisons is that, in the hand-crafted KB used previously, the KB author was able to specify objects that could be confused with the given object, along with other useful properties that might play a role in comparison. These potential confusors are not available in the automatically generated KB. We are still exploring the appropriate use of comparisons in this context, but see Milosavljevic and Oberlander (1998) for more information.

5.2 Multilinguality

A major change for the better is that the current system is fully multilingual, with descriptions available in English, French, Spanish, Dutch, and Chinese. The main difficulty with the multilinguality is that one of the major components of the database is the names of the objects themselves. Translating these names to each of the target languages is a daunting (but not impossible) task. Currently, the English names of items are always shown. Figure 4 shows the text of the description shown in Figure 3 in French and Spanish.

```
Ces objects sont des `Balmoral boots'. Ils ont ete fabriques en entre
1870 et 1875. Ils ont ete fabriques a `London'. Ils sont en
`leather', patent leather, glace kid, `linen' et `wood'. Les
`Balmoral boots' sont 45 mm de hauteur, 255 mm de longeur, 55 mm de
largeur total, 150 mm de hauteur total et 30 mm de largeur.
```

```
Estos son `Balmoral boots'. Fueron hechos entre los aÒos 1870 y 1875.
Fue producido en `London', Inglaterra. Est·n hechos de cuero, patent
leather, glace kid, lino y madera. Los `Balmoral boots' tienen 45 mm
de altura, 255 mm de largo, 55 mm de anchura total, 150 mm de altura
total y 30 mm de ancho.
```

Figure 4: French and Spanish descriptions of the Balmoral boots

For some of the smaller sets of lexical items (e.g., materials and places), translations have and are continuing to be made. These are presented to the user in the target language. Adding a new language is straightforward, requiring the translation of about 400 materials and countries and the translation of the template sentences used by the generator.

6. What's missing?

Clearly, we could improve the quality of the texts generated if we could do a better job of extracting information from the database of objects. Unfortunately, this is currently out of our hands. The experience would have been very different if the database had been more consistently marked-up with XML or another representation language.

The lack of typed links on the Web leads us to produce texts that are probably less fluent than they could be. If it were possible to specify exactly what discourse intention would be understood by the user following a particular link, then we could do away with the lists of links at the bottom of each text that we produce. This would appear to be possible under the current extended linking proposal for XLink.

One of the advantages of a hand-crafted KB is that it is a relatively straightforward task to specify possible links between objects while building it. Although we could use a similarity metric to propose links between items based on the KB features that they have in common, the sparseness of our current data would mean that most such comparisons would be based on size. It may be that we can resort to using IR-like techniques (see Green, 1997) to automatically determine which objects are related, but this will most likely only work for the textually rich objects. By the same token, this would allow us to make use of data that we are currently ignoring.

Finally, there is no straightforward (or perhaps integrated) way to get an overview of what the information space underlying the system is. There is currently no way for a user to get some idea of how many things there are to look at or how to get to them.

References

(Bernard, 1990) Bernard J. Ed. *The Macquarie Thesaurus*. Macquarie Library, North Ryde NSW. 1990.

(Dale et al., 1998) Robert Dale, Jon Oberlander, Maria Milosavljevic and Alistair Knott. Integrating natural language generation and hypertext to produce dynamic documents. *Interacting with Computers*. 11(2). 15 December 1998.

(Green, 1997) Stephen J. Green. Automated link generation: Can we do better than term repetition?. In *Proceedings of the Seventh International World Wide Web Conference*, April, 1998, Brisbane, Australia, pp. 75--84.

(Hitzman et al., 1997) Hitzeman J., Mellish C. and Oberlander J. Dynamic generation of museum web pages: The intelligent labelling explorer. *Archives and Museum Informatics*, 11:105--112. 1997.

(Milosavljevic et al., 1996) Maria Milosavljevic, Adrian Tulloch and Robert Dale. 1996. Text Generation in a Dynamic Hypertext Environment. In *Proceedings of the Nineteenth Australasian Computer Science Conference (ACSC'96)*, Melbourne, Australia. 31 January - 2 February 1996, 417-426.

(Milosavljevic and Oberlander, 1998) Maria Milosavljevic and Jon Oberlander. 1998. Dynamic Hypertext Catalogues: Helping Users to Help Themselves. In the *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, Pittsburgh, PA, USA, 20-24 June 1998.

(Reiter and Dale, 1999) Ehud Reiter and Robert Dale. 1999. *Building Natural Language Generation Systems*. Cambridge University Press.

A Key for Enhanced Hypertext Functionality and Virtual Documents: Knowledge

Philippe Martin and Peter Eklund

Griffith University, School of Information Technology,
PMB 50 Gold Coast MC, QLD 9726 Australia
Tel: +61 7 5594 8271; Fax: +61 7 5594 8066;
E-mail: {philippe.martin,p.eklund}@gu.edu.au

1. What the users want: precise and organized information, not documents

Web search engines - such as Altavista[1] or Infoseek[2] - retrieve entire documents based on the keywords they include. They exploit undirected Web robots to periodically traverse and index internet/intranet documents. Directed Web robots - such as Harvest[3], WebSQL[4] and WebLog[5] - apply string-matching and structure-matching commands (e.g. hypertext path expressions) to explore an intranet or a small subset of Internet and retrieve entire documents or parts of them. However, often people are not looking for lists of documents but either for *a precise answer* to a precise query, or for *a structured presentation of information related to a certain object* such as a particular event, technique, software type, idea or person. For example, someone looking for "large-scale deductive database systems" does not want a giant list of references to conferences, articles and courses on database systems, or home pages and user manuals of specific database systems, s/he first wants a classification of features that such systems may have, and then s/he may ask for a classification of existing tools according to some features, e.g. the kinds of query language, exploited techniques, API, memory & performance characteristics, support for multi-users, reliability, license.

Though such precise information and comparisons are important for each person interested in using deductive database systems, it is a long and difficult task for that person to collect the information just by reading documents. However, it is not necessarily difficult for each provider of information on an object to *represent* this information in a document or a shared knowledge repository *so that* they can be retrieved - and to a certain extent, merged or composed - via conceptual commands. As opposed to string-matching and structure-matching commands, conceptual commands rely on logical inferences (e.g. exploitation of subsumption relations between terms in the knowledge statements) and improve both precision and recall in information retrieval. They may also be combined with other commands within scripts or usual documents to create virtual documents.

2. Easing knowledge representation

The easiest way to express information is in natural languages. However, outside limited domains, these languages are too ambiguous for the semantic content of sentences to be automatically extracted. We argue in our article for the WWW8 conference [6]that general and intuitive knowledge representation languages or derived simpler notations (e.g. a "controlled language" that is a subset of natural language that eliminates sources of ambiguity) are preferable to metadata languages[7] based on XML[8] (e.g. RDF[9] and OML[10]) for indexing Web documents and representing knowledge within them. Indeed, the retrieval of precise information is eased by a language designed to represent semantic content and support logical inference, and the readability of such a language eases its exploitation, presentation and direct insertion within a document (thus also avoiding information duplication).

XML is intended as a machine-readable rather than human-readable language because it is mainly meant to be generated and read by machines not people. XML-based metadata

languages inherit this poor readability and most of them (e.g. RDF) do not specify how to represent logical operators or quantifiers. As an alternative, WebKB proposes to use expressive but intuitive knowledge representation languages to represent (or index) information in documents and mix knowledge statements with other textual elements (e.g. sentences, sections or references to images). To allow this, the knowledge (or commands exploiting it) must be enclosed within the HTML tags "<KR" and "</KR" or the strings "$(" and ")$". The knowledge representation language used in each chunk must be specified at its beginning, e.g.: "<KR language="CG"". (Lexical/structural/procedural commands may be used whichever language is specified). Thus, there is no need to separate knowledge from its documentation nor duplicate it in an external knowledge base.

At present,WebKB only exploits the CG (Conceptual Graph) formalism. However, the exploitation of wrappers (e.g. KIF to CGs) or other inference engines would allow WebKB to accept other knowledge representation languages. To compare the alternatives, here is an example showing how a simple sentence may currently be represented in WebKB, how it could be represented in KIF, and what its RDF representation is. The sentence is: "John believes that Mary has a cousin who has the same age as her".

```
<KR language="CG">
load "http://www.bar.com/topLevelOntology";
//Import this ontology Age
< Property; //Declare Age as a subtype of Property
Cousin(Person,Person) {Relation type Cousin}; //Declare relation
Cousin with its signature

[Statement: [Person: "Mary"]- { ->(Chrc)->[Age: *a];
                          ->(Cousin)->[Person]->(Chrc)->[*a];
                               }
]->(Believer)->[Person: "John"];
</KR>


<KR language="KIF">
load "http://www.bar.com/topLevelOntology";
//the WebKB command for file interpretation

(Define-Ontology Example (Slot-Constraint-Sugar topLevelOntology))
(Define-Class Age (?X) :Def (Property ?X))
(Define-Relation Cousin(?s ?p) "Relation type Cousin"
                    :Def (And (Person ?s) (Person ?p)))

(Exists ((?j Person))
  (And (Name ?j John)
    (Believer ?j '(Exists ((?m Person)(?p Person)(?a Age))
                          (And (Name ?m Mary) (Chrc ?m ?a)
                               (Cousin ?m ?p) (Chrc ?p ?a)
                          ))
         )))
</KR>


<!-- RDF notation (with allowed abbreviations);
     this file is named "example" -->
<RDF xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
     xmlns:t="http://www.bar.com/topLevelOntology">

  <Class ID="Age"> <subClassOf resource="t#Property"/> </Class>
  <PropertyType ID="Cousin">
     <comment>Relation type Chrc (Characteristic)</comment>
     <range  resource="t#Person"/>
     <domain resource="t#Person"/> </PropertyType>
</RDF>
```

```
<RDF xmlns="http://www.w3.org/TR/WD-rdf-syntax#"
     xmlns:t="http://www.bar.com/topLevelOntology"
     xmlns:x="http://www.bar.com/example">
<!-- x refers to this file -->


  <Description aboutEach="#Statement_01">
     <t#Believer>John</t#Believer>
  </Description>


  <t#Person bagID="Statement_01">
         <t#Name>Mary</t#Name>
         <x#Chrc><x#Age ID="age"></x#Age></x#Chrc>
         <x#Cousin><t#Person><x#Chrc resource="#age"/></t#Cousin>
  </t#Person>
</RDF>
```

The CG representation (top) seems simpler than the others. The semantic network structure of CGs (i.e. concepts connected by relations) has three advantages:

1. it restricts the formulation of knowledge without compromising expressivity and this tends to ease knowledge comparison from a computational viewpoint;
2. it encourages the users to express relations between concepts (as opposed, for instances, to languages where "slots" of frames or objects can be used);
3. it permits a better visualization of relations between concepts.

We advocate the use of Conceptual Graphs (CGs)[11] and simpler notational variants that enhance knowledge readability (e.g. we have also developed a formalized English and structured text notation). To further ease the representation process, we propose (i) a technique allowing users to leave some knowledge terms undeclared, and (ii) a top-level ontology of 400 concept and relation types. We have implemented a knowledge-based directed Web robot named WebKB to parse and execute our notations, knowledge handling & retrieval commands, Web document handling commands, and script language (to combine groups of commands). This tool is accessible as a CGI server. The WebKB site[12] provides HTML+Javascript interfaces.

Various kinds of applications of knowledge representation, indexation and queries are illustrated by examples in the WebKB site. Here is how some information on the Aditi database system could be represented in one of the structured text notations accepted by WebKB. The difference with the structured way (Information is extracted from the "Catalog of free database systems"[13]). Relations between each term used in this knowledge statement and other terms may be similarly defined elsewhere (in other documents or shared knowledge repositories) by one or several other users. Then, for example, subsumption relations between terms may be exploited for conceptual retrieval.

```
[Aditi.
  isa: large-scale deductive database system;
  user interface: NU-Prolog, graphical interface (implemented with: Motif);
  index method: B-trees, multi-level signature files;
  ports: SunOS, IRIX;
](representation date: 1992/12/17; representation author: aditi@cs.mu.oz.au).
```

3. Storing knowledge and commands in documents

It is handy for an information provider to store and structure knowledge inside Web documents, especially if the duplication of information into machine readable statements and human-only readable statements can be avoided (e.g. by using a controlled language[14] for sentences and a visual language[15] for graphics) or at least reduced by the possibility of

mixing and linking the two kinds of statements. To allow this, WebKB exploits the convention that each group of knowledge statements or commands in a document must be delimited by the two special HTML tags "<KR" and "</KR" or the strings "$(" and ")$". The knowledge representation language used in each group must be specified at its beginning, e.g.: "<KR language="CG"". Each group is visible unless the document's author hides it with HTML comment tags. Furthermore, various notations allow people to use knowledge statements for indexing any part of any Web document (not just parts which can be referred by URLs). Thus, knowledge statements may be retrieved and handled via document-based commands, and conversely indexed parts of documents may be retrieved and handled via knowledge-based commands.

When a command sent to the WebKB CGI server requires it to "run" a Web document (referred to by a URL), the server retrieves the document and executes the knowledge statements and commands within it (some commands may be to run other Web documents). The results are sent back to the client and constitutes a generated document (hence, a virtual document). Depending on a parameter, the WebKB server may or may not send back the human-only readable statements along with the results (if it does, the generated document is a copy of the original document with the query results in place of the commands). Similarly, the WebKB server may be used to exploit other CGI servers. Within HTML documents, dynamic linking may be achieved by using Javascript[16] to associate a command with an hypertext link in such a way that the command is sent to the WebKB server when the link is activated.

As with any other directed Web robot, the scalability and efficiency of the current WebKB is limited by the fact that (i) the users must know which documents contain (or may contain) the knowledge to exploit, and (ii) these documents must be accessed and parsed each time their content has to be exploited. Pieces of knowledge, like Web documents, may be provided by all Web users, and need to be inter-related or integrated, to allow each user to benefit from the knowledge of users they do not know. For this purpose a *cooperatively built knowledge repository* is necessary.

4. Storing knowledge and commands in distributed scalable knowledge servers

Some Web servers, called ontology servers, support shared knowledge repositories, e.g. the Ontolingua ontology server[17] and Ontosaurus[18]. However, they are not usable for managing large quantities of knowledge and, apart from AI-Trader[19], do not allow the indexation and retrieval of parts of documents. Finally, support of cooperation between the users is essentially limited to consistency enforcement, annotations and structured dialogues, as in APECKS[20], Co4[21] and Tadzebao[22].

We are extending WebKB to handle a *cooperatively built knowledge repository* which addresses scalability via the five following points[23]:

1. a scalable multi-user persistent object repository to support the storage and exploitation of knowledge structures (we have chosen the Shore[24] system);
2. algorithms allowing the exploitation of large-scale dynamic taxonomies efficiently (we have chosen Fall's algorithms[25]);
3. visualization techniques (mainly the handling of aliases for terms and the generation of views) to avoid lexical conflicts and enable users to focus on certain kinds of knowledge;
4. protocols to allow users to solve semantic conflicts via the insertion of new terms and relations in the common ontology and, in some cases, in the knowledge of other users;
5. conventions for representing knowledge to improve the automatic comparison of knowledge from different users and hence their consistency and retrieval.

Though these five points permit the exploitation of a large knowledge repository (essential

for efficiency reasons and practical use), it is also clear that for efficiency and reliability reasons, a unique server cannot be used to handle a universal knowledge repository by all Web users. Knowledge has to be distributed and mirrored on various knowledge servers. However, since there is no static conceptual schemas in knowledge bases, the techniques of distributed database systems - such as AlephWeb[26], Hermes[27], Infomaster[28] and TSIMMIS[29] - cannot all be reused.

A first step to the distribution of a knowledge repository is to duplicate it on several servers, with updates made on a server automatically duplicated in other servers. Some servers may be dedicated to searches and others to updates.

A second step is to have general servers and specialized servers. A specialized server would store the same knowledge as general servers plus knowledge related to a well-defined set of objects, e.g. knowledge expressed with the subtypes of certain types. Since these sets of objects are well-defined (extensively or via definitions), a general server would store the URLs of these servers and, when answering a query, delegate the query to the relevant servers if more precision is required. These sets of objects might be determined by the managers of specialized servers, or according to the frequency of accesses to objects in knowledge repositories. Whatever the specialized server a user updates, if the knowledge it enters is relevant to other servers (e.g. if the knowledge is expressed with general terms), it should be automatically duplicated in these servers. The rationale of all this duplication is to speed searches and simplify the query mechanisms by avoiding, whenever possible, parallel searches in various servers and then the composition of the results.

Other steps may be necessary, but what should be avoided in this knowledge-based (hence precision-oriented) approach is to let the specialized servers develop independently of one another instead of being part of a unique consistent virtual knowledge repository. Otherwise, conceptual queries and cooperation across the repositories are no more possible, and as in current traders, a most relevant repository to answer a query has to be automatically "guessed".

Finally, knowledge servers should not be limited to storing knowledge statements: they should also allow a storage and handling of knowledge-based and document-based commands similar to the storage and handling we described for documents.

5. Conclusion

The more a piece of information is precisely represented, the more adequately it can be retrieved and exploited. General and intuitive knowledge representation languages seem best adapted for this end. WebKB permits the use of Conceptual Graphs as well as simpler notations when less expressivity or precision is required. Ambiguities due to declared terms are partially solved according to the constraints in the used ontologies.

Storing knowledge within documents is useful but the scalability of this approach is limited. Ultimately, we believe a knowledge-based Web relies on scalable distributed cooperatively built knowledge repositories and automated knowledge acquisition techniques. We have proposed (and work on) some directions for this goal. In this view, knowledge-annotated documents are used as isolated modules of knowledge on which a user can work before submitting content to a knowledge server for integration. A document including commands can also be sent to a knowledge server as a template for generating virtual documents. Of course, scripts of commands could also be stored in a repository handled by a knowledge server and referred to from a document. We are currently extending WebKB to allow for these combinations of features.

In the same way we register a Web site today, we will probably register knowledge representations (or documents including knowledge representations) and complement or refine one another's knowledge.

References

1. Altavista: http://www.altavista.digital.com/
2. Infoseek: http://www.infoseek.com
3. Harvest: http://harvest.transarc.com/
4. WebSQL: http://www.cs.toronto.edu/~websql/
5. WebLog: http://www.cs.concordia.ca/~special/bibdb/weblog.html
6. WWW8 article: http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/papers/www8/www8.ps
7. Metadata languages: http://www.w3.org/Metadata/
8. XML: http://www.w3.org/XML/
9. RDF: http://www.w3.org/RDF/
10. OML: http://wave.eecs.wsu.edu/CKRMI/OML.html
11. CGs: http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/CGs.html
12. WebKB: http://meganesia.int.gu.edu.au/~phmartin/WebKB/
13. Databases: http://www.cis.ohio-state.edu/hypertext/faq/usenet/databases/free-databases/faq.html
14. Controlled languages: http://www-uilots.let.uu.nl/Controlled-languages/
15. Visual languages: http://www.cpsc.ucalgary.ca/~kremer/home.html\#visualLanguages
16. Javascript: http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm
17. Ontolingua ontology server: http://WWW-KSL-SVC.stanford.edu:5915/
18. Ontosaurus: http://www.isi.edu/isd/ontosaurus.html
19. AI-Trader: http://www.vsb.informatik.uni-frankfurt.de/projects/aitrader/intro.html
20. APECKS: http://www.psychology.nottingham.ac.uk/staff/Jenifer.Tennison/APECKS/
21. Co4: http://ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96b.html
22. Tadzebao: http://ksi.cpsc.ucalgary.ca:80/KAW/KAW98/domingue/
23. Repository ideas: http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/coopKBbuilding.html
24. Shore: http://www.cs.wisc.edu/shore/
25. Fall's algorithms: http://www.cs.sfu.ca/cs/people/GradStudents/fall/personal/index.html
26. AlephWeb: http://www.pangea.org/alephweb.aleph/paper.html
27. Hermes: http://www.cs.umd.edu/projects/hermes/
28. Infomaster: http://infomaster.stanford.edu/infomaster-info.html
29. TSIMMIS: http://www-db.stanford.edu/tsimmis/tsimmis.html

A Modular Framework for the Creation of Dynamic Documents

Jörg Caumanns
Free University of Berlin
caumanns@wiwiss.fu-berlin.de

Abstract

In this position paper a modular framework for the creation of dynamic documents is proposed. The motivation for this framework is to make use of the World Wide Web as both a source and a target for dynamic documents. To reach this goal we propose the use of an information store *as an intermediate layer between retrieval and integration of document fragments and semantic information. By hiding away retrieval from integration even existing dynamic document creating applications can make use of already available and forthcoming information retrieval and natural language processing systems.*

Introduction

Whenever a more or less famous person of our times and business enjoys the world with his visions about what the brave new networked hypermedia future will look like, the on-the-fly creation of documents is among the promises made [6, 10]. The idea of dynamically composing a document is indeed very exciting. It even gets more exciting if the resulting document not only exactly matches a user's request but even is adapted to his background and personality.

Most of these visions of dynamic documents are heavily influenced by the enormous growth of the World Wide Web. The Web is an ideal base for dynamic documents because it can be used as both source and target:

- The world's largest search engine - *Altavista* - claims to have indexed more than 100 million documents. These are 100 million information sources containing information to be extracted and reintegrated. These are 100 million documents consisting of a billion fragments to be split and restructured like the pieces of a jigsaw. These are 100 million documents covering a million of topics and providing an uncountable number of information which could be extracted, analysed, and transformed into big knowledge bases.
- The World Wide Web has means and standards for document presentation (HTML, Java, ActiveX, etc.), document exchange (HTTP), document linkage (hyperlinks), and dynamic document creation (CGI, ASP).

Most dynamic document systems make intensive use of the Web's target functionality. Only few of theses systems makes use of the Web as a dynamic and large fragment and/or information source. The major drawback of ignoring the Web's source functionality is the restriction to a set of well defined knowledge domains for which proprietary structural and semantic information is available.

In this position paper we propose a generic framework for the creation of dynamic documents that allows for the integration of powerful (not only web based) information retrieval systems. By adding the dynamic retrieval of fragments, semantic and factual information, dynamic document systems can become useful tools for learning and information about any topic a user is interested in.

Retrieval vs. Integration

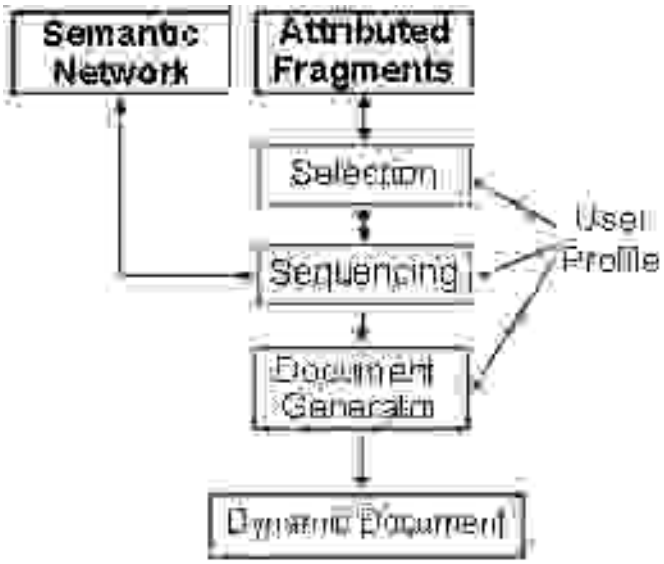


Figure 1 gives a coarse impression of the architecture of most dynamic document systems. The most important building blocks are

- some kind of semantic structure (e.g. a semantic or a conceptual network) providing information about the knowledge domain covered by the system,
- a collection of fragments (canned texts, phrases, images, etc.) needed to put together the surface of the resulting document,
- a selection module to select the most appropriate among the fragments,
- a sequencing module for discourse planning and structuring, and
- a document generator to integrate the selected and sequenced fragments or phrases into a set of HTML files.

Whether and how these building blocks are implemented depends on the kind of dynamic document system:

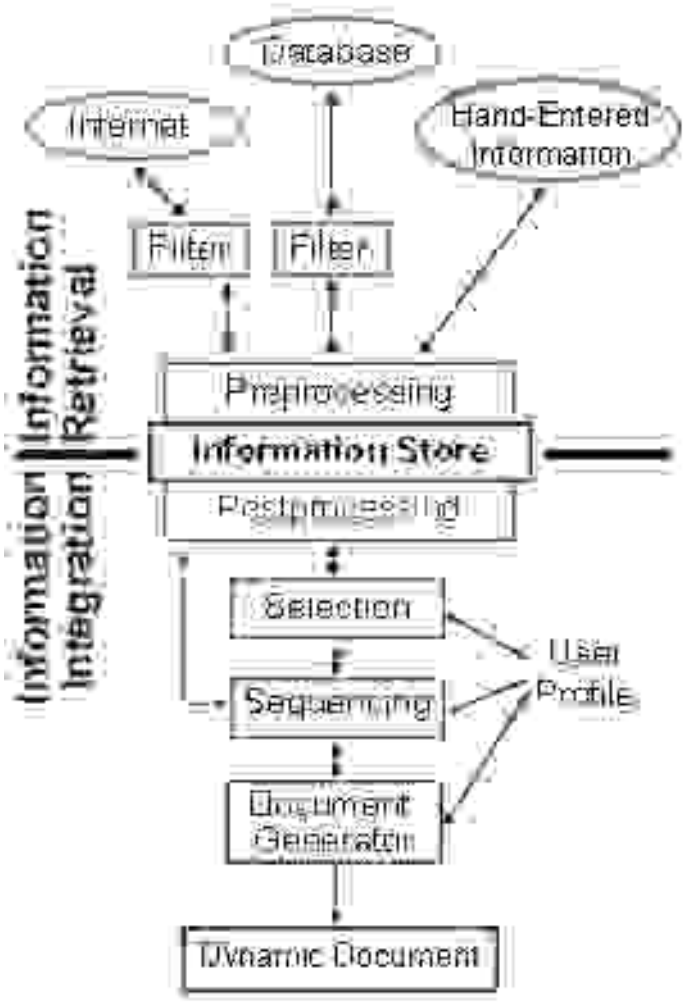
- With a natural language generating system like ILEX [8] the fragment store contains a lexicon of phrases. Sequencing is named *text planning* and makes use of a *network of facts*. The document generator is implemented as a *surface realisation component*.
- Fragment based bottom-up oriented systems (non-planning approaches doing selection first) like i4 [3] don't require a semantic network. Instead the semantic information needed for selection and sequencing is kept with each fragment. This causes the demand for the fragment store to hold both fragments and meta-data (attributes in this case). The selection module picks the most appropriate among the fragments and passes them to the sequencing module to create the discourse structure. Integration of fragments into HTML code is again task of the document generator.
- Fragment based top-down approaches (discourse planning adaptable hypertext systems) like ELM-ART [2] use a simple conceptual network for discourse planning. For each concept that is part of the resulting document a providing fragment is selected. All selected fragments are sequenced and integrated into a set of HTML files.

All of these approaches have in common that at least the fragments and / or the semantics have to be provided manually:
"Hand-entered information includes type hierarchies for jewels and designers, ..." [8]
"This work has made use of an online encyclopedia." [7]
"The first step is to collect a set of mediaobjects and provide meta-information about them. This is nearly the only task that has to be performed manually." [4]
"A special graphical editor for concept structures allows the creation of concepts, connecting them among each other with various types of semantic relations ..." [12]

The major drawback of the demand for hand-entered information is the restriction to a well defined knowledge domain. E.g. if a user wants to learn something about climate on mars he has to rely on *Altavista*, because none of the dynamic document systems available contains semantic information and / or fragments about the solar system.

A second drawback is the fixed nature of the hand-entered data. Manual maintenance of a semantic network about a rapidly changing topic (e.g. the German tax system or the Italian government) is nearly impossible and will in the long run result in non-current documents. Further more many domains allow for different interpretations of facts (e.g. the Clinton-Levinsky case or some economic topics like the European Monetary Union). A static semantic structure can usually only reflect a single interpretation, which prevents an objective, balanced description of "facts".

To overcome these problems, the semantic structure as well as the document building fragments should be retrieved dynamically. By doing so arbitrary user requests could be answered with dynamic documents. Even rapidly changing and highly subjective topics could be handled because each time a user requests information about such a topic a new, current, and adapted semantic network would be set up.



The figure above shows how such a framework for adaptable, dynamic documents that makes use of dynamic fragment and semantics retrieval could look like. In NLG and top-down oriented scenarios, data from various sources is first collected and analysed to set up a semantic network. The semantic network is hidden within an information store to provide a layer of abstraction between retrieval, selection, and sequencing. Documents matching parts of the semantic network are retrieved, split, normalised and stored within

the information store, too. Based on these information discourse planning, selection, sequencing and document creation is done using any of the existing approaches. For a bottom-up oriented scenario without discourse planning, documents matching some domain specific queries are retrieved and analysed. Then these documents are split, indexed, and stored in the information store. In parallel a fragment graph is set up within the information store that is used for selection and sequencing. If a semantic network is available for the given knowledge domain, semantic document analysis can be omitted, because the fragment graph can as well be set up using this information.

Information sources may be accessed through filters. Possible filters for databases could be anything from query languages up to complex information integration systems for combining the contents of many different databases. Among filters for WWW based information sources could be search engines (e.g. *Altavista*), query languages (e.g. *WebSQL* [13]), clustering and classification systems (e.g. *SONIA* [11]), or any combination of these services.

Dynamic retrieval of semantic structures cannot be completely automated using current technology. For this reason hand-entered information can be added to the information store. By hiding hand-entered information inside the information store existing dynamic document systems can easily adapted to the proposed modular architecture.

This directly leads to the main idea behind the proposed architecture: Starting with only manually provided information the content of the information store is step by step attributed, enriched, and extended by automatically retrieved information and fragments. E.g. if a hand-entered semantic network is given, it should be possible to dynamically retrieve a high percentage of the documents, fragments, or phrases needed by making use of existing IR technology. Especially web sites already providing meta-information about their contents and structure (e.g. by using *Dublin Core* [5] or Web-Schemes like *Araneus* [1]) could be good sources for this kind of information. Further more, existing semantic structures can be used transparently by accessing them through the generic interface of the information store. Even if all semantic structures and fragments have to been entered manually, the framework supports the transparent integration of services like language detection, quality measuring, document fragmentation, etc.

Another advantage of this modular architecture is the possibility to logically and physically distribute the various services, e.g. by using a CORBA-like infobus architecture. Mapped to the CORBA terminology, selection, sequencing (discourse planning), and document creation would be *application objects*, the information store a *common facility*, and source wrappers, filters and additional services *common object services.* Communication between the services would be done through an *object request broker*.

The Information Store

The information store acts as an intermediate layer between the information retrieval and information / fragment integration part of a dynamic document system. Its main purposes are to

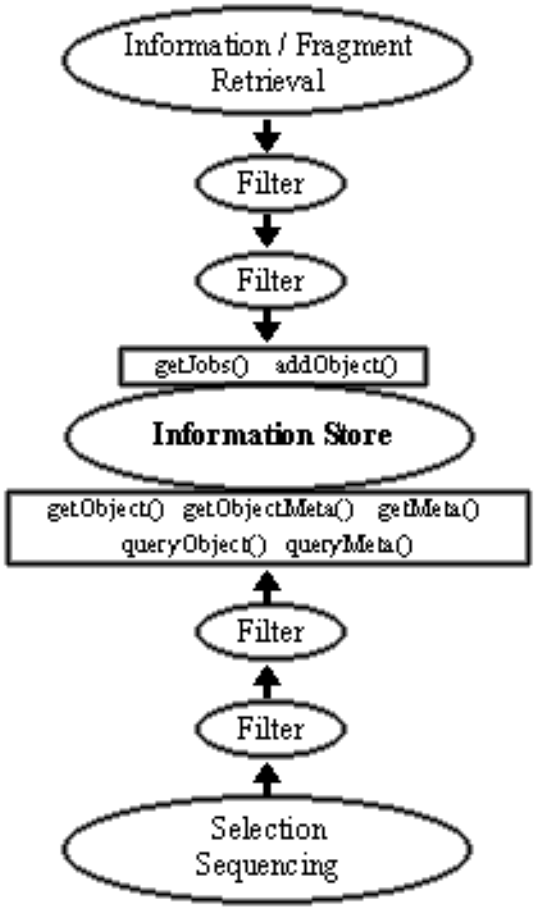
- decouple these layers from each other by providing an abstract interface for accessing semantic information as well as concepts, fragments or phrases
- enable the integration of services like language detection or quality measuring (e.g. [8])
- allow the externalisation of problems like pricing and authentication

The most important part of the information store is its interface. How semantic structures and fragments are stored (and even if they are stored at all) and how they are encoded internally is completely implementation dependant. The information store may either be implemented as a real store or just as a cache. If it is implemented as a cache it is just used as an abstraction layer, e.g. to wrap an existing semantic network or to integrate fragments from different sources.

The two kinds of objects maintained by the information store are fragments and concepts.

Fragments may be of any size and type ranging from single words or phrases up to large, mixed-type documents and digital video. Outside the information store each concept and fragment is just a set of name-value pairs (attributes). Relationships between any two objects (object mappings) can be assigned attributes as well in order to set up semantic networks, fragment graphs, and indices.

Interfaces



The figure above shows how the information store is used to separate information and fragment retrieval from integration. Various filters can be placed between the information store and the other building blocks either to provide additional functionality or to wrap interfaces. What filters are needed and used depends on the retrieval system, the additional services required, the dynamic document system, and the knowledge domain.

The information store should at least contain seven standardised interfaces - two on the retrieval side and five on the integration side:

getJobs()

Return a list of what kind of fragments, facts, or semantic information is currently required.

addObject()

Add an object or an object mapping to the information store. The content, encoding, price, and other attributes of the object must be described by attributes.

getObject()

Read an object or object mapping from the information store.

getObjectMeta()

Read only a certain attribute's value from the information store.

getMeta()

Get meta data about all objects. These meta data may range from the number of objects stored up to a conceptual network created dynamically from all objects within the information store.

queryObject()

Test whether a certain object or mapping exists within the information store. The description of the desired object should be based on meta data describing the object.

queryMeta()

Test whether a certain kind of global meta data can be provided by the object store.

The only purpose of the two query interfaces is money. The idea is that any call to one of the query interfaces is for free, while retrieving objects or meta data from the objects store is potentially not.

Meta Data

Each object or mapping within the information store is described by attributes. The values of these attributes can partly be provided by an information retrieval system, partly be calculated by various filters or the information store itself, and partly be set by hand.

What kind of meta data is available depends on the type of the object. E.g. concepts could be described by synonyms, grammatical and morphological rules, domain specific information, etc. Fragments may be attributed by

- their **contents** (e.g. required and provided concepts or a conceptual network encoded as a set of concept-concept mappings)
- **formal aspects** (e.g. type, size, age, language, etc.)
- **copyright information** defining by whom, for what purpose, and how the object may be used
- **authentication information** defining when, where and by whom the object was retrieved and how and by whom the originality of the object was proven (or can be proven).
- **pricing information** stating who has to pay what amount of money to whom for using the object.

Mappings of concepts to concepts are used to describe semantic or conceptual networks. Possible attributes could be the kind and direction of relationship between the two objects. Mappings of fragments to fragments are required by bottom-up oriented, fragment based systems in order to set up fragment graphs. For systems that do discourse planning based on semantic structures mappings from concepts to their providing fragments are needed. All of these mappings are mainly attributed by various weight specifiers, e.g. how a concept is explained by a certain fragment.

In order to make meta data as extensible and flexible as possible some of the attribute's names and semantics should be taken from existing standards (e.g. Dublin Core [5]), some have to be standardised, and some should be left open to the implementers of the retrieval and the integration parts.

Filters

The functionality of the information store can be extended by filters. Retrieval side filters are mainly used to calculate additional meta data while integration side filters provide additional services like pricing, type conversion, or formatting.

The main idea of providing the ability to add filters to the information store is to make use of already available systems (e.g. language detection, document fragmentation, stemming, keyword extraction) and to externalise open problems (e.g. pricing).

Acknowledgements

This work is supported by the German Research Network as part of the DIALECT/DIALERN project and by the German Research Society as part of the Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK316)

References

1. Araneus Homepage, http://www.dia.uniroma3.it/Araneus/index.html.
2. Brusilovsky, P., Schwarz, E., and G. Weber, ELM-ART: An intelligent tutoring system on World Wide Web. In *3rd International Conference on Intelligent Tutoring Systems, ITS-96*, Montreal, June 1996.
3. J. Caumanns, "A Bottom-Up Approach to Multimedia Teachware", In *4th International Conference on Intelligent Tutoring Systems, ITS-98*, San Antonio, 1998.
4. Caumanns, J. and H-J. Lenz, "Hypermedia Fusion - A Document Generator for the World Wide Web", In *IEEE Multimedia Systems 99, ICMCS99*, Fierence, June 1999 (*to appear*).
5. Dublin Core Home Page, http://purl.oclc.org/dc/.
6. B. Gates, *The Road Ahead*, Viking Penguin, New York, 1995.
7. Hearst, M., Kopec, G., and D. Brotsky, "Research in Support of Digital Libraries at Xerox PARC." Available as http://www.dlib.org/dlib/june96/hearst/06hearst.html.
8. Milosavljeciv, M. and J. Oberlander, "Dynamic Hypertext Catalogues: Helping Users to Help Themselves." In *9th ACM Conference on Hypertext and Hypermedia, HT'98*, Pittsburgh, June 1998. Available as http://www.cmis.csiro.au/Maria.Milosavljevic/papers/ht98/.
9. Naumann, F., Leser, U., and J.C. Freytag, *Quality-driven Integration of Heterogeneous Information Sources*, Technical Report HUB-IB-117, February 1999. Available as http://www.dbis.informatik.hu-berlin.de/~naumann/HUB-IB-117.ps.gz
10. N. Negroponte, *Being Digital*, Knopf, New York, 1995.
11. Sahami, M., Yusufali, S., and M.Q.W. Baldonado, "SONIA: A Service for Organizing Networked Information Autonomously." In *Third ACM Conference on Digital Libraries, DL98*, Pittsburgh, June 1998.
12. J. Vassileva, "Dynamic Course Generation on the WWW", In *Workshop on Intelligent Educational Systems on the World Wide Web*, Kobe, August 1997.
13. WebSQL Home Page, http://www.cs.toronto.edu/~websql/

The value-adding functionality of Web documents

Kevin Crowston[1] and Marie Williams[2]

[1] Syracuse University
School of Information Studies
4-206 Centre for Science and Technology
Syracuse, NY 13244-4100
crowston@syr.edu
Phone: +1 (315) 443-1676
Fax: +1 (315) 443-5806

[2] Web Architechs
206 Meadowbrook Dr.
Syracuse, NY 13210
williams@web-arch.com
Phone: +1 (315) 426-0272
Fax: +1 (315) 426-0679

The World-Wide Web (or the Web) is an Internet client-server communication system for retrieving and displaying multi-media hypertext documents (Berners-Lee, et al., 1994). Documents are identified by an address called a Uniform Resource Locator or URL. The Web's main advantage over earlier Internet systems is its merger of retrieval and display tools, its capacity for handling formatted text, embedded graphics and other media and point-and-click links to other documents (hence the name). As well, many browsers are capable of seamlessly retrieving information using older protocols (e.g., FTP, Gopher and Usenet News) and automatically launching other applications to display diverse Internet data types (e.g., sound, animation).

As a basis for studying organizational communications, Yates and Orlikowski (1992; Orlikowski and Yates, 1994) proposed using genres. They defined genres as, "typified communicative actions characterized by similar substance and form and taken in response to recurrent situations" (Yates and Orlikowski, 1992, p. 299). They further suggested that communications in a new media would show both reproduction and adaptation of existing communicative genres as well as the emergence of new genres.

Crowston and Williams (in press) found numerous examples of pages that recreated genres familiar from traditional media. They also saw examples of genres being adapted to take advantage of the linking and interactivity of the new medium and novel genres emerging to fit the unique communicative needs of the audience.

More interesting, the technology of the Web enables novel applications based on a shift from static documents to "live" data. For example, Yan et al. (1996) describe how patterns of user access can be used to suggest which information should be viewed next. However, these uses of the Web are unlike conventional documents. Instead, we expect such systems to recreate genres from information systems. Therefore, we propose using a typology of system value-adding processes to categorize the functionality of Web sites. Specifically, we propose using Taylor's value-added model (Taylor, 1986). Value-added processes are those "characteristics or attributes which are added to the data and information items being processed that make them more useful to users than they were at the start of the process" (Taylor, 1986, p. 19). An information system adds value by helping users make choices or clarify their options. In this framework, many uses of the web are such value-adding systems.

Taylor goes on to list numerous ways that a system can be value-adding (what he calls "user criteria of choice"): by enhancing ease of use (e.g., easier access to data), by noise reduction (e.g., providing selected information), by quality (e.g., accuracy or currency), by adaptability (e.g., addressing a specific user need) and by time or cost savings.

For example, a Web search engine such as Altavista provides ease of use by allowing

searches for documents with specified keywords. On the other hand, a Web index such as Yahoo provides noise reduction, since it includes only a selection of Web sites and may improve quality, if the selection is accurate, comprehensive, current and reliable. An article database such as ABI/Inform or Lexis/Nexis goes even further in providing noise reduction (e.g., by providing abstracts of articles) and quality (e.g., by selecting articles from trusted sources). Finally, hot lists of sources (e.g., the MkLinux resources page) provide what Taylor calls adaptability, because the sources chosen are those that are directly relevant to a specific task (in this case, setting up a computer to use MkLinux). All of these systems may provide time or cost savings compared to retrieving the information by taking a trip to the library (though perhaps at the expense of one of the other criteria).

Bibliography

Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F. and Secret, A. (1994). The World-Wide Web. *Communications of the ACM*, 37(8), 76-82.

Crowston, K. and Williams, M. (in press). Reproduced and emergent genres of communication on the World-Wide Web. *The Information Society*.

Orlikowski, W. J. and Yates, J. (1994). Genre repertoire: The structuring of communicative practices in organizations. *Administrative Sciences Quarterly*, 33, 541-574.

Taylor, R. S. (1986). *Value Added Processes in Information Systems*. Norwood, NJ: Ablex Publishing Corporation.

Yan, T. W., Jacobsen, M., Garcia-Molina, H. and Dayal, U. (1996). From user access patterns to dynamic hypertext linking. In *Fifth International World Wide Web Conference*. Paris, France.

Yates, J. and Orlikowski, W. J. (1992). Genres of organizational communication: A structurational approach to studying communications and media. *Academy of Management Review*, 17(2), 299-326.

Automated Hypermedia Support for the Virtual Documents Generated by Analytical Applications

Michael Bieber, Roberto Galnares

New Jersey Institute of Technology
{bieber, galnares}@njit.edu; http://www-ec.njit.edu/~bieber

Motivation

Our domain are the everyday analytical applications used daily in the real world [BK95]. By "analytical applications" we mean computational applications such as database management systems, decision support systems, spreadsheets and other systems that generate their display content in real time, often as a result of user queries and other interaction. The documents and display screens are virtual - they often do not exist before being displayed. Therefore hypermedia constructs must be added dynamically, in real time, as the document is about to be displayed. Furthermore, links cannot be inferred only using lexical analysis. Does the number "6" in a spreadsheet represent a month or net income?

Background

We are developing the DHymE hypermedia engine to provide automated hypermedia support to analytical applications. DHymE uses mapping rules, which provide links for a particular type of object. For example, if a document contains a value representing the net income for a company, DHymE would search for mapping rules for "net income" objects. Our goal is supplementing analytical applications with hypermedia support, without altering them. We are developing the RA relationship analysis approach [BY99] to help developers determine which relationships are in their application domain. Each of these can then be specified by a mapping rule, and therefore become mapped to a hypermedia link..

This paragraph gives a technical description of DHymE's interaction with an analytical application. The DHymE hypermedia engine executes separately from the target application. We write a wrapper program for each application to integrate it into our engine architecture. Applications or their wrappers then connect to DHymE through a Web proxy server. DHymE intercepts all messages passing between the application and the user interface, and uses the mapping rules to map each appropriate element of the message to a hypermedia node or anchor. Our Web browser wrapper integrates these anchors into the document being displayed and passes it through the proxy server to the user's Web browser. When the user selects an anchor, the browser wrapper passes it to DHymE, which returns a list of possible links (one for each appropriate relationship as determined by the mapping rules). If the user selects a normal application command (mapped to an operation link), DHymE passes the command on to the application for processing. If the user selects a hypermedia engine link (e.g., to create an annotation), DHymE processes it entirely. If the user selects a supplemental relationship, DHymE infers the appropriate application commands, meta-application operations (e.g., at the operating systems level or schema level) or hypermedia engine operations that will produce the desired information. If the user selects a user-created annotation, DHymE retrieves it. Thus DHymE automatically provides all hypermedia linking (as well as navigation) to applications, which remain hypermedia-unaware and in fact often entirely unchanged. We currently are integrating several applications with DHymE, automatically giving each a Web interface or supplementing its existing Web interface: a personnel requisition tracking system, a relational database management system, a spreadsheet, and a mathematical model management system. [Bi99] describes these ideas and an older, non-Web prototype of

DHymE in more detail.

Issues

There are many interesting issues involving virtual documents when mapping hypermedia to analytical applications. Here are a few:

*** Parsing Virtual Documents**

How do we parse the virtual documents created by analytical applications to determine where the elements of interest are? If an element of interest has a mapping rule for its element type, then the DHymE hypermedia engine creates a link anchor over that element of interest corresponding to the mapping rule.

*** Document Specifications**

How does one specify the set of commands that have to be passed to an application to create a specific document or screen? What parameters will be needed to specify a particular instance? These commands must be written in a general way and embedded in a mapping rule. It then must be instantiated with the particular instance that the user wants to see, based on which link anchor the user has just selected.

*** Regenerating Virtual Documents**

Suppose the user creates a bookmark or a stop along a guided tour to a virtual document or screen. Then the document or screen is closed. How do we regenerate that document or screen later, especially if the user had to input parameter values to create it in the first place? (We don't want to ask the user to re-input these parameter values.)

*** Target Areas**

How do we specify specific arrival anchors (target areas) within virtual destination documents once a link has been traversed and the virtual document is about to be displayed?

*** Saving Virtual Documents**

Who owns a virtual document - the user or the application? If the user saves it, do we strip out the hypermedia components we added to the document for integrity purposes? Should it be stored centrally by the DHymE engine or locally on the user's disk? May it be altered by the user or would the system creator wish to keep it intact for legal or copyright reasons?

and the old standard problems:

*** Link and Annotation Integrity**

If the content of the document changes, how do we relocate links?

*** Link and Annotation Relevance**

How do we know when a link or annotation has lost its relevance?

References

[Bi99] Bieber, M. "Supplementing Applications with Hypermedia," under revision for ACM Transactions on Information Systems. [On-line: http://www.cis.njit.edu/~bieber/pub/supp/supp.html]

[BK95] Michael Bieber and Charles Kacmar, "Designing Hypertext Support for Computational Applications," Communications of the ACM 38(8), 1995, 99-107.

[BY99] Bieber, M. and J. Yoo, "Hypermedia: A Design Philosophy," in submission. [On-line: http://www.cis.njit.edu/~bieber/pub/ht-philosophy/phil.html]