

Aspects of Semantic Data Modeling in Hypermedia Systems

Denis Helic

Aspects of Semantic Data Modeling in Hypermedia Systems

Dissertation for the Award of the Academic Degree
Doctor of Technical Sciences
at
Graz University of Technology

submitted by

Denis Helic

Institute for Information Processing and Computer Supported New Media (IICM)
Graz University of Technology
Austria

July 2001

© Copyright 2001 by Denis Helic

First Reader: O.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer
Second Reader: Ao.Univ.-Prof. Dr. Nick Scherbakov

Aspekte von semantischem Datenmodellieren in Hypermedialen Systemen

Dissertation zur Verleihung des akademischen Grades
Doktor der Technischen Wissenschaften
an der
Technischen Universität Graz

vorgelegt von

Denis Helic

Institut für Informationsverarbeitung und Computergestützte neue Medien (IICM)
Technische Universität Graz
Österreich

Juli 2001

© Copyright 2001, Denis Helic

Diese Dissertation ist in englischer Sprache verfasst.

Erster Begutachter: O.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer
Zweiter Begutachter: Ao.Univ.-Prof. Dr. Nick Scherbakov

Abstract

The World Wide Web is a vast global distributed hypermedia system providing an access to large amount of hypermedia information. This thesis analyzes the possibilities to support different knowledge transfer processes on the Web. Such processes might improve facilities for managing or retrieving hypermedia information on the Web. To analyze such possibilities this thesis looks on the Web from the data modeling point of view.

Thus, the thesis identifies the logical, physical and semantic data model of the Web. Further, the thesis shows that in order to support a number of different knowledge transfer processes we must apply novel semantic data modeling approaches to the Web.

The thesis presents three such novel semantic data models: HC-Data model, Knowledge Domains and Knowledge Cards, which support Web-based knowledge structuring, Web-based knowledge profiling and Web-based knowledge mining, respectively.

HC-Data model extends the well-known concept of hypermedia composites allowing authors to define appropriate roles for members of their hypermedia composites. Composites created in this way represent well-defined chunks of structured knowledge in a hypermedia form.

Knowledge Domains provide profiled global overviews of hypermedia resources in Web applications. Such overviews consist of a number of semantic entities interrelated by means of semantic relationships. In order to create these overviews, hypermedia resources might be assigned to these semantic entities and their relationships. Accessing hypermedia information structured by means of Knowledge Domains is achieved through semantic browsing and structured semantic querying.

Knowledge Cards extend the concept of Knowledge Domains by exploiting important properties (infer possibility) of such semantic data structures. Thus, accessing a Knowledge Card results in accessing both: assigned hypermedia resources as well as inferred hypermedia resources. This greatly facilitates an initial access to best-match hypermedia resources on the Web.

The thesis presents the implementation of these semantic data modeling approaches in a Web environment.

Finally, the thesis provides the overview of related work and suggests few ideas for further development.

Kurzfassung

Das World Wide Web ist ein riesiges globales verteiltes hypermediales System, das einen Zugriff auf eine riesige Menge von hypermedialen Information gestattet. Diese Dissertation analysiert die Möglichkeiten zur Unterstützung von verschiedenen Wissenstransferprozessen auf dem Web. Solche Prozesse könnten Möglichkeiten zu einer verbesserten Verwaltung und Retrieval von hypermedialen Information bieten. Um eine solche Analyse durchzuführen, wird in dieser Dissertation das Web aus einer Datenmodellierungsperspektive untersucht.

Daher identifiziert diese Dissertation das logische, das physische und das semantische Datenmodell vom Web. Weiteres zeigt diese Dissertation dass um eine Anzahl von verschiedenen Wissenstransferprozessen auf dem Web zu unterstützen, innovative semantische Datenmodellierungsansätze angewandt werden sollen.

Diese Dissertation präsentiert drei solche innovative semantische Datenmodelle: HC-Data Model, Knowledge Domains und Knowledge Cards, die das Web-basiertes Wissensstrukturieren, das Web-basiertes Wissensprofilieren und das Web-basiertes Wissens-Mining unterstützen.

HC-Data Model erweitert das bekannte Konzept von hypermedialen Kompositen indem es Autoren ermöglicht angemessene Funktionen für Kompositenteile definieren zu können. Die auf dieser Weise erstellten Kompositen stellen ein strukturiertes Wissen in einer hypermedialen Form dar.

Knowledge Domains bieten profilierte globale Übersichten von hypermedialen Ressourcen in Web Applikationen. Solche Übersichten bestehen aus einer gewissen Anzahl von semantischen Kategorien, die untereinander mit semantischen Relationen verknüpft sind. Um solche Übersichten zu erstellen, ordnet man hypermediale Ressourcen zu Kategorien und deren Relationen zu. Der Zugriff auf die auf diese Weise strukturierte hypermediale Information erfolgt durch semantisches Navigieren und strukturiertes semantisches Suchen.

Knowledge Cards erweitern das Knowledge Domain Konzept indem man einige wichtige Eigenschaften (Inferenzmöglichkeit) von solchen semantischen Datenstrukturen zu nutzen versucht. Daher bedeutet das Zugreifen auf eine Knowledge Card zweierlei: das Zugreifen auf die zu dieser Knowledge Card zugeordneten hypermedialen Ressourcen, sowie das Zugreifen auf durch die Inferenz gewonnenen hypermedialen Ressourcen. Dadurch ist ein erstes schnelles Zugriff auf geeignete hypermediale Ressourcen auf dem Web gewährleistet.

Diese Dissertation präsentiert auch die Implementierung von den vorgestellten semantischen Datenmodellierungsansätzen in einer Web-basierten Umgebung.

Zum Schluss bietet die Dissertation ein Überblick von anderen relevanten Arbeiten auf diesem Gebiet sowie einige Ideen für mögliche zukünftige Entwicklung.

I hereby certify that the work presented in this thesis is my own and that work performed by others is appropriately cited.

Ich versichere hiermit, diese Arbeit selbstständig verfasst, andere als angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Acknowledgments

Everyone at the IICM has been ready to provide me with valuable help, comments and feedback. Above all, I would like to thank Hermann Maurer and Nick Scherbakov for providing me with ideas and suggestions for the work presented in this thesis.

Denis Helic
Graz, Austria, July 2001

Table of Contents

1	Introduction	1
2	Overview of Logical Hypermedia Data Modeling	3
2.1	Hypermedia systems and WWW	3
2.1.1	The definition of hypertext and hypermedia	3
2.1.2	A short history of hypermedia	4
2.1.3	Hypermedia systems	6
2.2	World Wide Web	7
2.2.1	Navigating WWW	8
2.2.2	Creating and modifying a hyperweb on the Web	9
2.3	Data modeling in general	10
2.3.1	The definition of database, data model and different levels of data modeling	10
2.4	Data model of the Web	12
2.4.1	Physical data model of the Web	12
2.4.2	Logical data model of the Web	12
2.4.3	Process sub-model of the Web	13
2.4.4	Data sub-model of the Web	13
2.4.5	Semantic data model of the Web	13
2.5	Advantages of the node-link data model	14
2.6	Disadvantages and problems of the node-link data model	14
2.6.1	Insufficient structuring	14
2.6.2	Lack of meta-data	15
2.6.3	Logical integrity	15
2.6.4	Limiting link philosophy	16
2.6.5	Non-contextual links	16
2.6.6	The "lost in hyperspace" syndrome	17
2.6.7	Poor search hit rates	17
2.6.8	Inadequate authoring support	17
2.6.9	Unsatisfactory reuse of hypermedia material	20
2.6.10	Insufficient support for collaborative work	21
2.6.11	Inconvenient security systems	21
2.6.12	Weak connection to databases	21
2.7	Current trends in logical hypermedia data modeling	21
2.8	HM-Data model	22
2.8.1	Data structures	22
2.8.2	Browsing	23
2.8.3	Data classes	25
2.8.4	Creating and modifying a hyperweb in the HM-Data model	25
2.8.5	Introducing the HM-Data model to the WWW	26
2.9	Discussion of the logical composites approach	28
2.9.1	Advantages	28
2.9.2	Disadvantages	29
2.10	Limitations of logical hypermedia data modeling	30
2.10.1	Knowledge structuring	31
2.10.2	Knowledge profiling	32
2.10.3	Knowledge mining	33
3	Semantic Hypermedia Data Modeling	35
3.1	Semantic data modeling in general	35
3.2	HC-Data model: Knowledge structuring in hypermedia systems	38
3.2.1	HC-Data model	38
3.2.2	Data structures	39
3.2.3	Defining properties of a new HC-Type	40
3.2.4	Defining navigational structure of a new HC-Type	41
3.2.5	Browsing	42
3.2.6	Creating and modifying a hyperweb	43
3.2.7	Mapping the HC-Data model onto the Web	43
3.2.8	An application of the HC-Data model	44
3.2.9	Illustrative example	45

3.2.10	HC-Types in WBT-Master	46
3.2.11	Discussion of the semantic hypermedia composites approach	47
3.3	Knowledge Domains: Knowledge profiling in hypermedia systems	48
3.3.1	Knowledge representation in hypermedia systems	48
3.3.2	Knowledge Domains	49
3.3.3	Data structures	50
3.3.4	Defining a Knowledge Domain Schema	51
3.3.5	Creating and modifying a database	52
3.3.6	Browsing	53
3.3.7	Searching	54
3.3.8	Discussion of the Knowledge Domains concept	54
3.4	Knowledge Cards: Knowledge mining in hypermedia systems	56
3.4.1	Properties of semantic networks	56
3.4.2	Knowledge Cards	57
3.4.3	Creating and modifying a hyperweb	58
3.4.4	Browsing	58
3.4.5	Inference engine	59
3.4.6	Discussion of the Knowledge Card approach	60
4	Introducing Semantic Data Structures to the WWW	61
4.1	WBT-Master	61
4.1.1	WBT-Master architecture	61
4.1.2	WBT-Master technical solutions	62
4.1.3	WBT-Master GUI	63
4.2	HC-Data model in WBT-Master	64
4.2.1	Working with HM-Linker	64
4.2.2	HC-Units Navigation Panel	66
4.3	Knowledge Domains in WBT-Master	70
4.3.1	Working with Knowledge Domain Schema authoring tool	70
4.3.2	Working with Knowledge Domain Content administration tool	74
4.3.3	Browsing a Knowledge Domain	75
4.4	Knowledge Cards in WBT-Master	76
4.4.1	Knowledge Card Control Panel	76
4.4.2	Knowledge Card Administration Panel	77
5	Related Work	79
5.1	Hypermedia systems utilizing semantic data modeling	79
5.1.1	Interbook	79
5.1.2	HyperTutor	79
5.1.3	PEGASUS	80
5.1.4	Ontobroker	80
5.2	Resource Description Framework	80
5.2.1	The Semantic Web	80
5.2.2	Knowledge Domains on the Semantic Web	81
5.2.3	Example of mapping Knowledge Domain onto RDF	82
5.2.4	Knowledge Cards on the Semantic Web	84
5.2.5	Example of mapping Knowledge Cards onto RDF	84
5.3	Topic Maps	86
5.3.1	Topic maps and Knowledge Domains	87
5.3.2	Topic maps and Knowledge Cards	87
6	Conclusion	89
	Bibliography	91

List of Figures

Figure 2.1 Hypertext.....	3
Figure 2.2 Hypermedia.....	4
Figure 2.3 Documents, links and anchors in hypermedia.....	4
Figure 2.4 Hyperwave collection hierarchy.....	6
Figure 2.5 Distributed hypermedia system.....	7
Figure 2.6 Basic WWW concepts.....	8
Figure 2.7 WWW architecture.....	8
Figure 2.8 Browsing WWW.....	9
Figure 2.9 Standard HTML authoring tool.....	10
Figure 2.10 Data model.....	10
Figure 2.11 ANSI/SPARC database management system architecture.....	11
Figure 2.12 Sub-models of the node-link data model.....	12
Figure 2.13 "Spaghetti" links.....	14
Figure 2.14 Inaccessible document.....	15
Figure 2.15 Dangling links.....	16
Figure 2.16 Non-contextual links.....	17
Figure 2.17 Simple hyperweb.....	18
Figure 2.18 Authoring of hyperweb.....	19
Figure 2.19 Resulting hyperweb after modifications.....	20
Figure 2.20 Limitations of material reuse.....	20
Figure 2.21 Mapping internal data model onto WWW data model.....	22
Figure 2.22 Internal structure of S-Collection.....	23
Figure 2.23 Browsing current container.....	23
Figure 2.24 Zoom-In operation.....	24
Figure 2.25 Navigational planes.....	24
Figure 2.26 Types of S-Collections.....	26
Figure 2.27 Authoring an S-Collection.....	27
Figure 2.28 Rendering an S-Collection (the menu becomes a current container).....	27
Figure 2.29 Rendering an S-Collection (the member "Hardware" is accessed).....	28
Figure 2.30 HM-Data Model vs. Node-Link Data Model.....	29
Figure 2.31 Knowledge structuring: Defining new composite types.....	32
Figure 2.32 Knowledge profiling: Identifying categories and relationships.....	33
Figure 2.33 Knowledge mining: Identifying concepts in a hypermedia database.....	34
Figure 3.1 Entities and relationships.....	35
Figure 3.2 Typical ER-diagram.....	37
Figure 3.3 HC-Type, member roles and navigational structure defined with a DDL.....	38
Figure 3.4 HC-Type and HC-Units.....	39
Figure 3.5 Person HC-Type and an HC-Unit.....	39
Figure 3.6 Properties of Person HC-Type.....	41
Figure 3.7 Defining Screen Template topology.....	42
Figure 3.8 Navigational structure of Person HC-Type.....	42
Figure 3.9 Authoring an HC-Unit.....	44
Figure 3.10 Rendering an HC-Unit of Person HC-Type.....	44
Figure 3.11 Sample hypermedia database.....	46
Figure 3.12 Semantic network of resources in a hypermedia database.....	49
Figure 3.13 Semantic categories and their instances.....	49
Figure 3.14 Semantic relationships.....	50
Figure 3.15 Knowledge Domain schema.....	50
Figure 3.16 Categories, data items and relationships.....	51
Figure 3.17 Data manipulation processes supported by a schema.....	52
Figure 3.18 Creating an instance of a semantic category.....	53
Figure 3.19 Browsing an instance of a semantic category.....	53
Figure 3.20 Searching in a Knowledge Domain.....	54
Figure 3.21 Resulting hypermedia database.....	55
Figure 3.22 Different applications of semantic networks.....	57
Figure 3.23 Interrelating knowledge cards into a semantic network.....	58
Figure 3.24 Resources attached to a Knowledge Card.....	58

Figure 3.25 Browsing Knowledge Cards	59
Figure 3.26 Resources automatically inferred for a Knowledge Card.....	59
Figure 4.1 WBT-Master client-server architecture.....	62
Figure 4.2 WBT-Master server architecture.....	63
Figure 4.3 WBT-Master functional panel.....	64
Figure 4.4 HM-Linker	65
Figure 4.5 Setting member parameters with HM-Linker	66
Figure 4.6 Course Navigation Panel.....	67
Figure 4.7 Course Map.....	68
Figure 4.8 Course Table of Content	68
Figure 4.9 Help Window.....	69
Figure 4.10 Selecting “Look and Feel” for a particular course	69
Figure 4.11 Knowledge Domain Schema Functional Panel.....	70
Figure 4.12 Defining a new Knowledge Domain Schema	71
Figure 4.13 Defining categories and relationships	71
Figure 4.14 Defining a new semantic category	71
Figure 4.15 Schema with defined categories.....	72
Figure 4.16 Defining a category item.....	72
Figure 4.17 Defining a new relationship	73
Figure 4.18 The complete schema.....	73
Figure 4.19 Previewing the defined schema.....	73
Figure 4.20 Knowledge Domain Content Administration Functional Panel.....	74
Figure 4.21 Adding a training resource as a category instance	74
Figure 4.22 Defining attributes and installing relationships for a new category instance	75
Figure 4.23 Accessing all instances of a particular category.....	75
Figure 4.24 Previewing a category instance.....	75
Figure 4.25 Browsing a category instance.....	76
Figure 4.26 Knowledge Card Control Panel	77
Figure 4.27 Knowledge Card Administration Panel.....	77
Figure 4.28 Defining a new Knowledge Card.....	78
Figure 4.29 Adding a Learning Resource.....	78

List of Tables

Table 2.1 Time-line of hypermedia development.....	5
Table 2.2 Process Sub-Model.....	13
Table 2.3 Data Sub-Model	13

1 Introduction

The World Wide Web started as a small Internet based hypertext project at CERN (European Organization for Nuclear Research) in late 1990. In the past ten years the Web matured to the largest distributed hypermedia system. Now, the Web with its millions of servers, pages and users constitutes the largest library of human knowledge mankind has ever created.

In this thesis, I look on the Web from the data modeling point of view. Hypermedia systems in general and the Web in particular may be seen as a special kind of database management systems. Thus, they commit to a certain data model, at the physical, logical and semantic level.

The logical data model of the Web is called the node-link data model. This model supports the basic structuring of data into nodes and links and the basic information retrieval technique known as browsing. In early days of the Web development the primitive node-link data model was the cause of a number of the well-known problems, such as tedious authoring, link inconsistency, lack of overall structure, to mention just a few. A number of different achievements at the field of the logical hypermedia data modeling, such as hypermedia composites, resolved many of those Web problems.

Despite the fact that the Web overcame these early problems it still has a number of serious limitations. Considering the Web as the largest *library of human knowledge*, the Web still does not support efficiently, if at all, a number of well-known *knowledge transfer* processes. For instance, accessing a relevant information chunk on the Web, which constitutes so-called *knowledge mining* process, is a rather difficult task to accomplish by means of the current Web technologies. A further example involves structuring of a Web application according to a set of application-specific data structures with explicitly defined roles within that application. Such process may be seen as *knowledge structuring* process. Also, this process is not yet supported by means of the Web data models. Finally, the process of so-called *knowledge profiling*, i.e., the process of identifying different semantic categories and relationships between those categories and, as a next step, assigning hypermedia resources to these categories and relationships is not supported at all in the Web today.

I claim that the logical hypermedia data modeling paradigms are not sufficient to support these knowledge transfer processes on the Web. Rather an introduction of semantic hypermedia data modeling elements is required. This fact has lead me to development of a number of semantic hypermedia data modeling approaches that support the knowledge structuring, knowledge profiling and knowledge mining process on the Web.

The rest of this thesis is organized as follows: Chapter 2 gives an overview of the logical hypermedia data modeling paradigms. It starts with the definition of the terms used throughout this document. Thus, the terms such as hypertext, hypermedia, hypermedia systems, distributed hypermedia systems, the Web and data model are defined. Further, this chapter describes and discusses the logical data model of the Web – the node-link data model. Afterwards, current trends in logical hypermedia data modeling are presented, most notably the concept of so-called hypermedia composites. The chapter is concluded with a discussion on limitations of the logical hypermedia data modeling in general. Here I define the above mentioned

knowledge transfer processes and claim that such processes may only be supported by an introduction of semantic data modeling paradigms for the Web.

Chapter 3 presents the main work at the field of the semantic hypermedia data modeling for the Web that I did in the past few years. It describes the three different semantic data modeling approaches, namely the HC-Data model, Knowledge Domains and Knowledge Cards that support the knowledge structuring, knowledge profiling and knowledge mining processes, respectively.

Chapter 4 presents the WBT-Master, a novel Web-based-training system that implements the above-mentioned semantic data modeling concepts. This chapter deals with implementation issues of such concepts in a Web environment.

Chapter 5 gives the overview of related work, such as Resource-description-framework (RDF) or Topic-maps initiatives.

Finally, Chapter 6 provides a conclusion of the work and possible suggestions for further development.

2 Overview of Logical Hypermedia Data Modeling

This chapter provides the overview of the most important logical data modeling paradigms in hypermedia systems in general and World Wide Web in particular. An in-depth analysis of the primitive node-link data model of the Web is given. Advantages, disadvantages and limitations of the node-link data model are presented. Some current trends in logical hypermedia data modeling, such as the hypermedia composites approach are presented as well. I conclude this chapter with a discussion of limitations of logical hypermedia data modeling in general and claim that introducing the elements of semantic hypermedia data modeling can solve such limitations.

2.1 Hypermedia systems and WWW

2.1.1 The definition of hypertext and hypermedia

As opposed to the typical printed book, where the text is read sequentially from the beginning to the end, hypertext offers a nonlinear access to the text. A typical hypertext consists of a number of chunks of textual information, usually called hyper nodes or simply nodes. Such nodes are interrelated by means of computer navigable links. Usually, links are denoted as highlighted phrases within hypertext nodes.

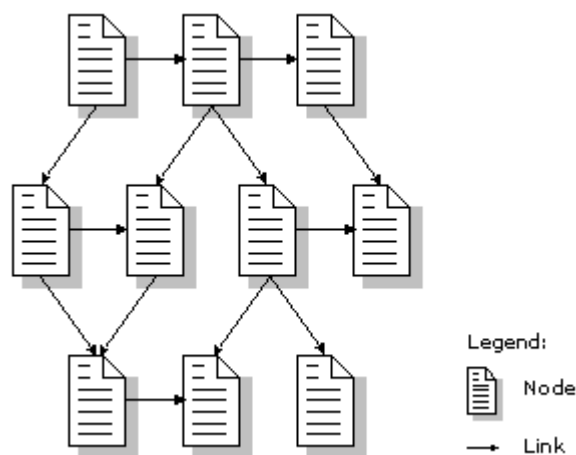


Figure 2.1 Hypertext

In hypertext, readers are not confronted with a prescribed reading sequence as it is the case with a book text, but rather they browse hypertext nodes activating computer navigable links. The nature of hypertext encourages readers to explore offered information intuitively, by association, following always to their most recent interests.

Hypermedia is a generalization of hypertext. In hypermedia nodes are multimedia nodes, i.e., they include not only textual information, but also other kinds of media, such as: digital images, sound clips, video clips

and similar. In a sense, hypermedia combines hypertext and multimedia paradigm into a new one. Often, hypermedia is referred to as the concept of imposing a navigable structure on the top of existing collection of multimedia nodes [Maurer and Scherbakov, 1996; Maurer et al., 1998]. As Conklin states it [Conklin, 1987]: *hypermedia is a style of building systems for organizing, structuring and accessing information around a network of multimedia nodes connected together by links.*

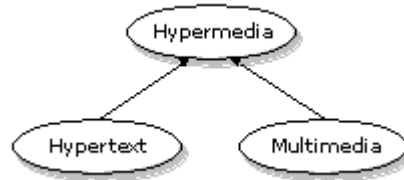


Figure 2.2 Hypermedia

A source anchor is the starting point of a link and specifies a part of a node where the link may be activated. Usually, source anchors are visualized in a special way (i.e., a piece of text may be highlighted) to notify users of the existence of a link. Similarly, a destination anchor is the ending point of a link and specifies a part of a node, which should be visualized upon the link activation. Usually, an entire node is specified as the destination anchor, but a specific part of a node may be specified as the destination anchor (e.g. a paragraph within a textual node) as well.

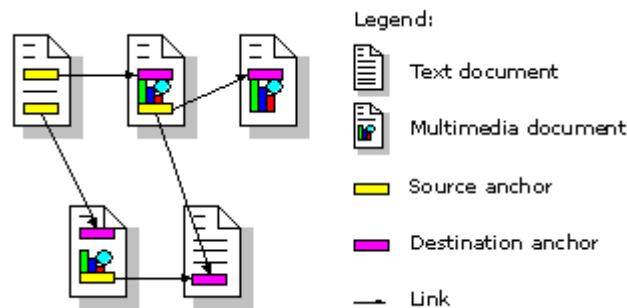


Figure 2.3 Documents, links and anchors in hypermedia

According to the basic structuring of data into nodes and links this basic hypermedia model is referred to as the node-link data model. Virtually, all other hypermedia models and systems may be found in this basic hypermedia data model. Likewise, a large part of hypermedia research assumes the underlying existence of this basic model [Rivlin et al., 1994]. According to this model, nodes and links in hypermedia systems form a directed graph network. Normally, such a network is called hyperweb or hyperspace.

In the early days of hypermedia development a hyperweb consisted of nodes containing mostly static multimedia information. Browsing such a hyperweb meant merely retrieving simple non-interactive multimedia documents. Recently, with the development of more advanced user interaction facilities hypermedia systems usually provide users with nodes containing interactive movies, virtual reality, collaboration and communication facilities, structured discussion forums and similar.

2.1.2 A short history of hypermedia

In his famous paper "As We May Think", published in 1945, Vannevar Bush described a system called "Memex" [Bush, 1945] (as for "memory expander"). Bush's "Memex" is a system that is very close to what we would call a hypertext system nowadays. One of the main ideas presented in this paper is the idea of an associative memory that one may access rather intuitively by following one's current associations. Bush considered that idea as being of primary importance. Also, in this paper he introduced the idea of so-called "trails". Trails link documents together, either sequentially or branched. "As we may see", analogies to modern hypertext systems are obvious. Because of such ideas many good surveys of hypertext and hypermedia systems place Bush as being the first person that contributed to development of hypermedia [Nyce and Kahn, 1991; Conklin, 1987; Nielsen, 1990; Tomek et al., 1991; Lennon, 1997].

Table 2.1 Time-line of hypermedia development

Year	System	Author
1945	Memex	Vannevar Bush
1960	Xanadu	Theodor Nelson
1968	On-Line System	Douglas Engelbart
1969	Hypertext Editing System	Andries van Dam
1978	Aspen Movie Map	Andrew Lipmann
1987	NoteCards	Frank Halasz
1988	Intermedia	Yankelovich
1989	Hypercard	Bill Atkinson
1990	Guide	Peter Brown
1991	World Wide Web	Tim Berners-Lee
1991	Hyper-G	Maurer
1994	HM-Card	Scerbakov

Another visionary, who coined the both terms "hypertext" and "hypermedia" was Theodor Nelson. In 1960 Nelson started with the work on the project called Xanadu [Nelson, 1965]. Basically, the Xanadu system is a document management system. The Xanadu system was based on library and publishing paradigms such as backtracking and versioning of documents. Nelson was the first person to implement the concept of the so-called "transclusion". This concept describes the function of including chunks of text from the linked destination document into the body of a document that contains the link's source anchor. Nelson considered this function as being of primary importance. The current developments in hypertext and hypermedia systems show that Nelson was right in this respect.

The first fully functional hypertext system was developed at Brown University, by the team around Andries van Dam and Ted Nelson [Carmody et al., 1969]. The system was called Hypertext Editing System, shortly HES. The system provided users with the possibility to browse and edit documents using a special function keyboard and lightpen. Moreover, the system supported two types (in-text links and branches) of links and annotations.

At the Stanford Research Institute, Douglas Engelbart and his team developed the system called NLS (oN-Line System) [Engelbart, 1963; Engelbart and English, 1968] in 1968. The NLS system is probably the first simple implementation of a hypertext system. Over 100,000 documents were stored in the system and users could browse those documents by following links between them. Beside offering cross-referenced information in the form of documents interrelated by means of computer navigable links, the NLS system offered structured information as well.

The Aspen Movie Map developed at the Massachusetts Institute of Technology Media Laboratory by Andrew Lippman and his group in 1978 [Lipmann, 1980] is the first linked multimedia system ever. The system incorporated two displays, one for a map and one for video clips of streets and buildings in Aspen, Colorado. Users were able to navigate through the map or point directly at a location on the map. The video corresponding to the clicked location was shown in the other display.

Another very innovative system was "Intermedia" [Yankelovich et al., 1988] developed at Brown University's Institute for Research in Information Scholarship by Yankelovich and his team. The main contribution of the Intermedia system to the hypermedia community was the introduction of an easy to use graphical user interface for navigation of the linked media.

Peter Brown at the University of Kent developed the first commercial hypertext system for a range of personal computers [Nielsen, 1990]. The system was called "Guide". Guide is both, an authoring and a browsing tool. It was comparatively easy to use.

NoteCards [Halasz, 1988], developed at Xerox PARC, by Frank Halasz and his group is the system that has been widely used commercially. The system was designed to help researchers and designers organize and develop their ideas [Lennon, 1997]. It is based on the idea of rectangular NoteCards containing individual data chunks. The system offers authoring tools for creating and/or editing NoteCards. Browsing is supported

by means of so-called browser cards. Browser cards provide a clickable map of the underlying structures (other NoteCards). Beside being the creator of the NoteCards system, Halasz contributed to the hypertext and hypermedia community with his milestone 1988 paper called "Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems" [Halasz, 1988]. This paper provided the definitive analysis of both the problems overcome and the issues still facing hypermedia system development.

In 1987, Bill Atkinson designed the hypermedia system called Hypercard [Hypercard, 1989]. The Hypercard system was bundled with every new Macintosh sold from 1987 until 1992. This fact opened for the first time an interactive multimedia authoring and browsing system to the general public. The HyperCard system provided users with an extremely simple graphical user interface to handle graphics, video, sound and interactive links to other cards.

The next major milestone in the history of hypermedia systems was start of the World Wide Web [Berners-Lee et al., 1992; Berners-Lee et al., 1992a; Berners-Lee et al., 1994], which was first demonstrated at the ACM 1991 Hypertext conference. The real breakthrough of the WWW started in 1993 after the National Center for Supercomputing Applications [NCSA, 2001] released their point-and-click graphical Web browser called Mosaic [Mosaic, 1993]. From this point in only a couple of years the WWW emerged to the largest distributed hypermedia system containing billions of documents.

At the approximately same time as the WWW was introduced, the group around Hermann Maurer and Nick Scerbakov at Graz University of Technology, was working on a system called Hyper-G [Kappe, 1991; Kappe, 1993; Maurer, 1996] (now commercially available as HyperWave [HyperWave, 2001]) and HM-Card [Maurer et al., 1994; Maurer et al., 1994a; Maurer et al., 1996, Maurer and Scherbakov, 1996]. The major contribution of these systems is the introduction of powerful data modeling facilities and hypermedia data structures such as collections, collection hierarchies or hypermedia composites.

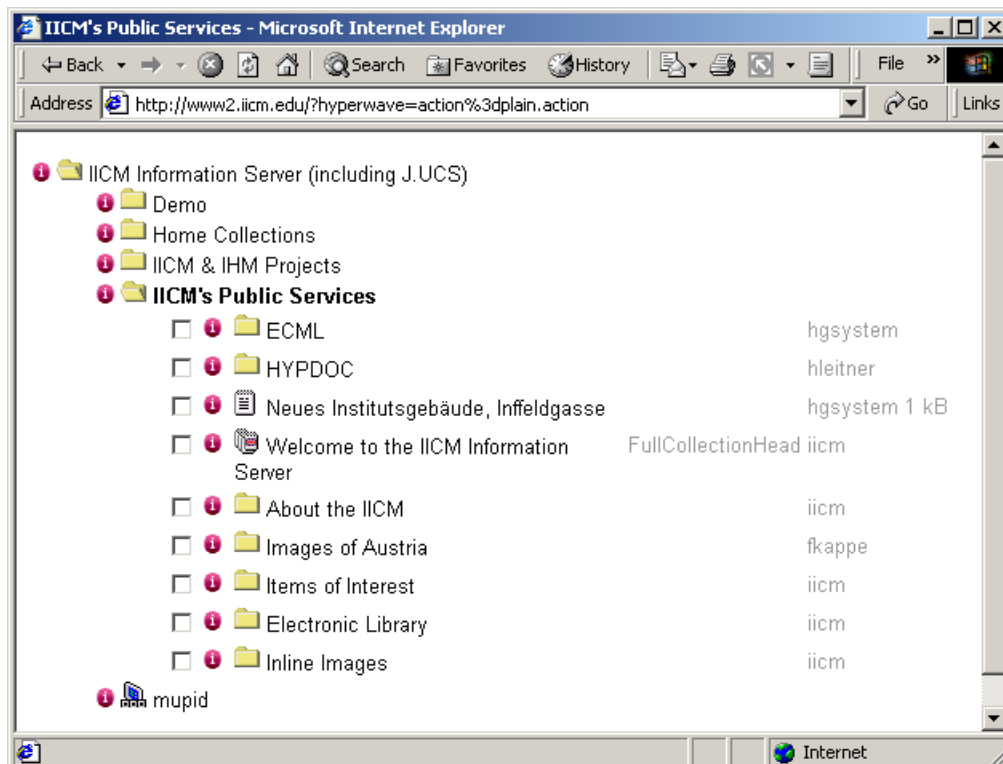


Figure 2.4 Hyperwave collection hierarchy

2.1.3 Hypermedia systems

We can classify hypermedia systems according to a number of criteria. For instance, considering the criteria of how much control over the form of the hypermedia presentation authors should have we get the following classification:

- Frame based hypermedia systems (all documents must fit into a fixed size frame)
- Window based hypermedia systems (documents may be of any size and they are displayed in a scrollable window area).

There are a number of other useful classification criteria for hypermedia systems. However, we consider the following classification as being of primary importance. According to the network environment in which hypermedia systems exist we distinguish between:

- Standalone hypermedia systems that provide access to a hyperweb residing on a single computer
- Large multi-user systems providing multi-point access to a hyperweb distributed over many computers connected in a network.

The focus of this work lies in the discussion of distributed hypermedia systems. A distributed environment is typically needed if either the information to be stored is too large for on machine, or if there are too many users for one machine to handle, or both. Obviously, the main prerequisite for distributed hypermedia systems is the existence of a distributed computer environment, i.e., a computer network.

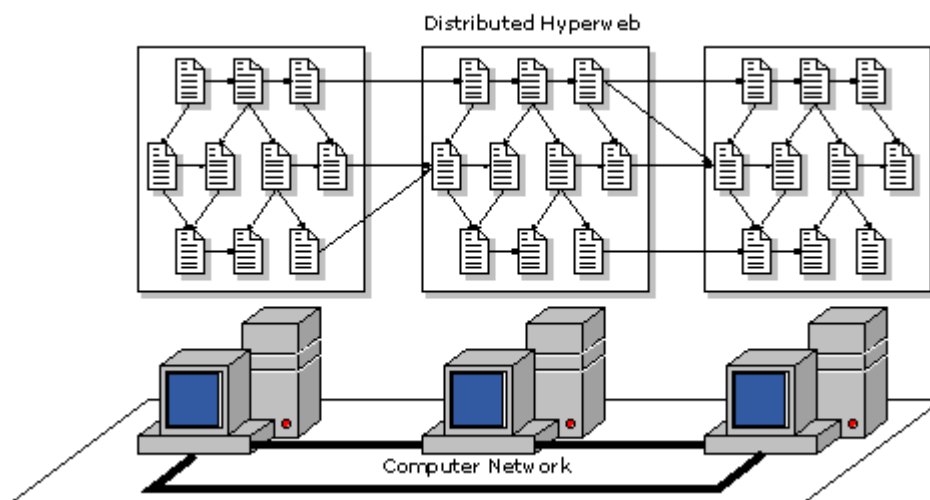


Figure 2.5 Distributed hypermedia system

Nowadays, the Internet is the largest worldwide distributed computer environment existing. It is in fact a network of networks that is estimated to connect several million computers and over 50 million individual users around the world - and it is still growing rapidly.

The Internet provides its users with a number of so-called Internet services such as e-mail service, file transfer service, remote login services and similar. However, the best-known and mostly used Internet service is the World Wide Web (WWW or Web). The World Wide Web is an Internet wide distributed hypermedia system. As such it is the largest hypermedia system existing.

2.2 World Wide Web

In this section I provide the overview of the main features of the World Wide Web from the users point of view.

The World Wide Web (the WWW or the Web) is the largest distributed hypermedia system nowadays. The WWW provides a remote access to the largest repository of hypermedia documents worldwide. The Web started at CERN in Geneva by Tim Berners-Lee and his team in 1991. However, the real breakthrough of the Web engaged upon the development of the first widely used Web browser, called Mosaic. Mosaic was developed by the National Center for Supercomputer Applications (NCSA). This browser provided a powerful graphical user interface following the simple point-and-click paradigm. Suddenly, by simply using Mosaic all documents residing on the Web were just a mouse-click away from users. This fact helped the

Web to become the most popular hypermedia system ever leading to a real explosion in numbers of Web sites and documents offered on the Web.

Speaking more technically, the Web is based on typical client/server architecture. In the WWW, the whole system functionality is distributed over a tremendous number of computers that are interconnected by means of the Internet. The WWW utilizes HTTP [HTTP, 2001] (HyperText Transfer Protocol) for client-server communication and HTML [HTML, 2001] (HyperText Mark-up Language) as a data exchange format. A particular Web server can be simply seen as a storage space for HTML and other kinds of documents where all such documents are accessible by means of so-called Uniform Resource Locator (URL) [URL, 2001].

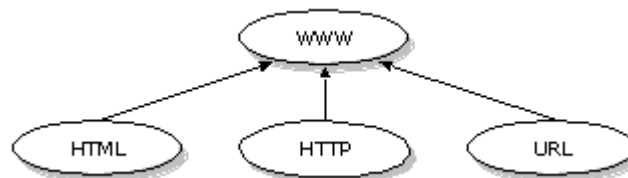


Figure 2.6 Basic WWW concepts

WWW servers accept HTTP requests and reply, usually, with an HTML document. Note that a URL is encapsulated into such HTTP request as a particular document identity. WWW clients just access HTML documents residing on WWW servers and visualize such documents on the user screen.

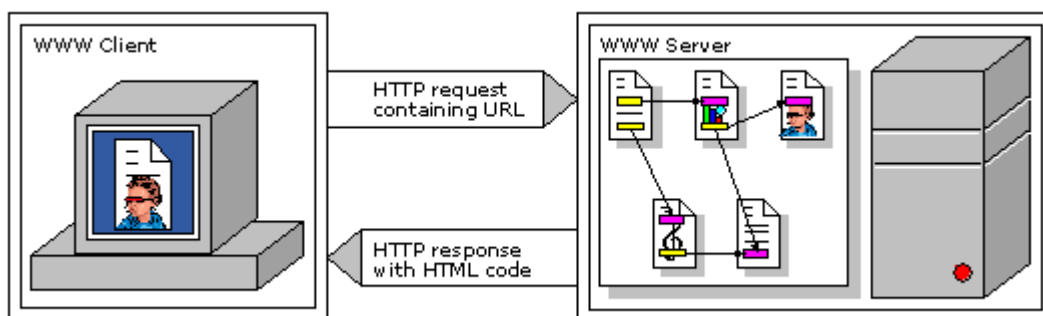


Figure 2.7 WWW architecture

Links in WWW are embedded into HTML documents as special tags containing a URL of a document, which is referred to. WWW clients are able to visualize and activate a hyperlink upon a user's action, thus requesting the linked HTML document to be visualized.

Alternatively to HTML documents WWW servers may store other kinds of documents such as: Windows Word [WinWord, 2001] documents, PowerPoint [PowerPoint, 2001] presentations, documents in Portable Document Format (PDF) [PDF, 2001], documents in eXtensible Markup Language (XML) [XML, 2001] format, etc. Recently, many of these document formats allow for embedding of hyperlinks, thus providing means for creating a hyperweb. However, in all such documents links are embedded into the documents, thus reducing the discussion of these formats on the discussion of the HTML format.

2.2.1 Navigating WWW

In order to retrieve information from the Web users operate a Web browser. A standard Web browser, such as Microsoft Internet Explorer [Explorer, 2001], or Netscape Navigator [Navigator, 2001] utilizes a point-and-click style of interaction with users. Thus, for basic information retrieval facilities in WWW users do not need to formulate queries. This fact makes hypermedia systems in general and the Web in particular easy to use and acceptable for millions of users.

However, users of hypermedia systems use a number of other information retrieval techniques, as well [Canter et al., 1985; McAleese, 1989; Andrews et al., 1995; Andrews, 1996]:

- Scanning: covering a large area without depth
- Browsing: following a path by association
- Searching: formulating queries to find an explicit goal
- Exploring: finding out the extent of the information space
- Wandering: ambling in an unstructured manner.

Nevertheless, on the Web, the basic information retrieval technique is still browsing. Usually, users browse the Web in an associative manner.

Many WWW sites offer a number of navigational tools to help users to navigate through the information space. These navigational tools include searching facilities, site maps, guided tours, hierarchical structures and similar. Recently, conceptual maps of WWW sites become increasingly popular (e.g., topic maps, resource site summaries, semantic hypermedia composites).

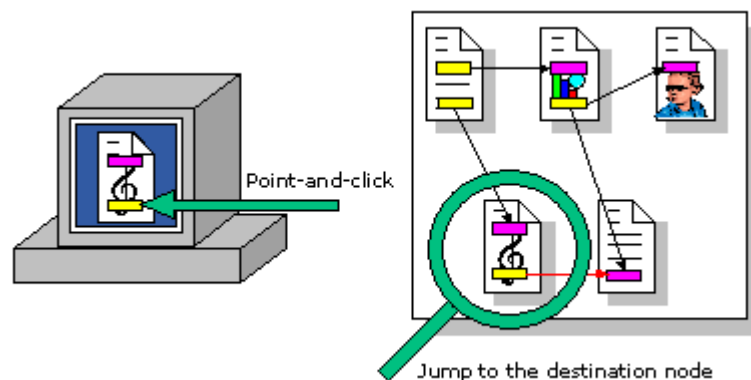


Figure 2.8 Browsing WWW

2.2.2 Creating and modifying a hyperweb on the Web

Usually, the process of creating and modifying a hyperweb in the WWW is referred to as authoring. The simplicity of the authoring process as found in the Web is another important fact that made the rapid growth of the WWW possible. Usually, in order to create a hyperweb authors create a number of HTML documents and interrelate them by means of computer-navigable links. Also, authors may include other type of documents into their hyperweb by simply referring to these documents.

HTML as a markup-language has the following important properties:

- It is very easy to understand even for inexperienced users – creating few HTML pages with a text editor is actually an easy-to-accomplish task.
- HTML is a de facto standard for data interchange on the Web – there exist not only a number of different powerful Web browsers, but also a wide range of HTML authoring tools such as Microsoft FrontPage [FrontPage, 2001], Netscape Composer [Composer, 2001], Macromedia Dreamweaver [Dreamweaver, 2001], CofeeCup HTML Editor [CofeeCup, 2001], to mention just a few. It is even easier to create a hyperweb by means of such editors than to write down the markup in a text editor.

HTML provides authors with a large number of different tags to define the content, structure and presentation of a hyperweb. Recently, Cascading Style Sheets (CSS) [CSS, 2001] are becoming increasingly popular for defining the properties of documents' presentation such as: fonts and their styles, font sizes, coloring schemas, spacing, margins, etc.

Usually, standard HTML editors are provided with facilities that allow authors to upload created documents on a WWW server. Thus, publishing of a created hyperweb on the WWW is as simple as clicking the "Upload" button.

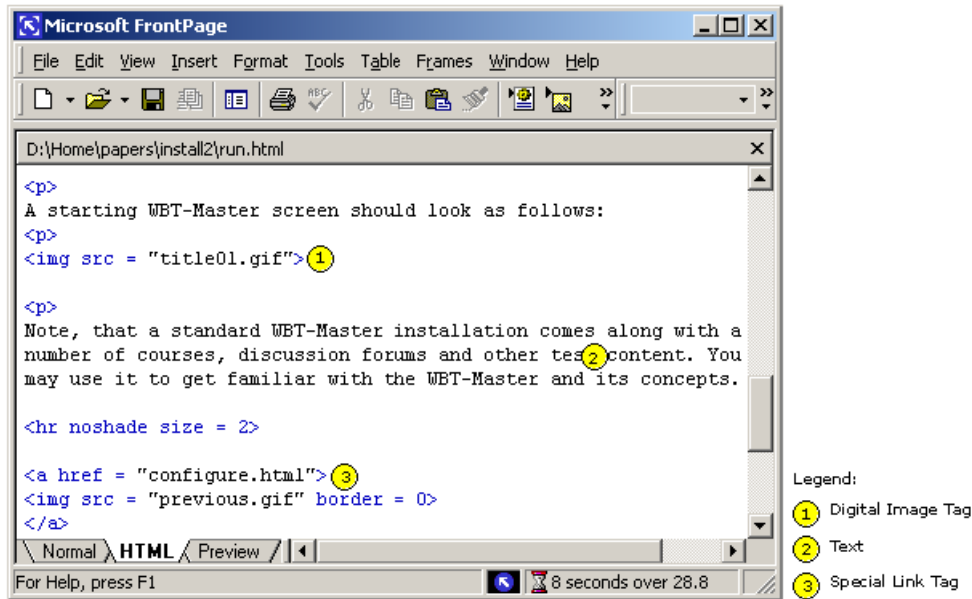


Figure 2.9 Standard HTML authoring tool

2.3 Data modeling in general

In order to be able to analyze the Web from a more technical point of view, i.e., from the data modeling point of view I will define a number of terms important for the further discussion. Thus, the terms such as database, data model and different levels of data modeling in a database management systems are defined in this section.

2.3.1 The definition of database, data model and different levels of data modeling

In an informal sense, a database is a set of data we create and maintain for reference and analysis. In a technical sense, a database imposes certain well-defined structures and operations on a data set so that creating, finding, analyzing, updating and maintaining the consistency of data is made as easy and as efficient as possible, particularly for very large data sets. In both senses, hypermedia systems are database applications [Maurer et al., 1998; Nielsen, 1990].

Each database management system utilizes a certain data model. A data model is defined to be a consistent, formal set of statements about how data can be organized and processed [Tsichritzis and Lochovsky, 1982]. It is composed of a defined set of elements and structures along with valid operations that can be performed on those elements and structures. E.F Codd [Codd, 1970] defines a data model as a combination of a data structure, operations and integrity rules.

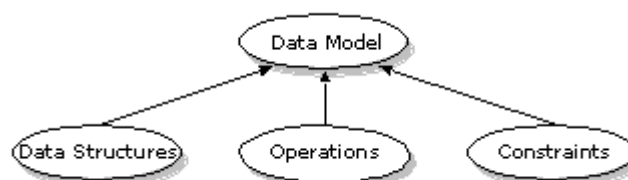


Figure 2.10 Data model

According to the terms from the ANSI/SPARC architecture of database systems [Tsichritzis and Klug, 1978] we distinguish between three levels of data modeling:

- Logical data model
- Physical data model
- Semantic data model.

In its most general sense, a logical data model defines application independent (i.e. generic) data structures and operations, which can be applied to basic data constructs. We can also define data model as the users' perception of the functionality of information system as such [Maurer et al., 1998].

For example, the well-known relational data model defines such functionality of a database system as a set of operations (relational algebra) or rules of inference (relational calculus), which can be applied to a predefined number of two-dimensional tables usually called relations [Maurer et al., 1998].

The second level, the physical data model deals with implementation issues of a logical data model on a specific hardware and software platform. For instance, a physical data model may define a special set of operations of how to implement a WWW site on a UNIX operating system environment.

The third level, the semantic data model (the conceptual or the meta level) specifies application-dependent or purpose-oriented data constructs. For instance, this level may define application-specific semantic data templates to simplify manipulation of logical data structures and to ensure the logical integrity of the whole database. Usually, templates provide data with so-called meta-data (i.e., data about data) that allow for semantic categorizing of data. Another very important aspect of semantic data modeling is so-called conceptual modeling. Here, we try to identify semantic entities (concepts) and semantic relationships (relations) between such entities. For instance, the data categories that are created by means of meta-data may be interrelated by means of semantic relationships, which identify different real-life relations between these categories.

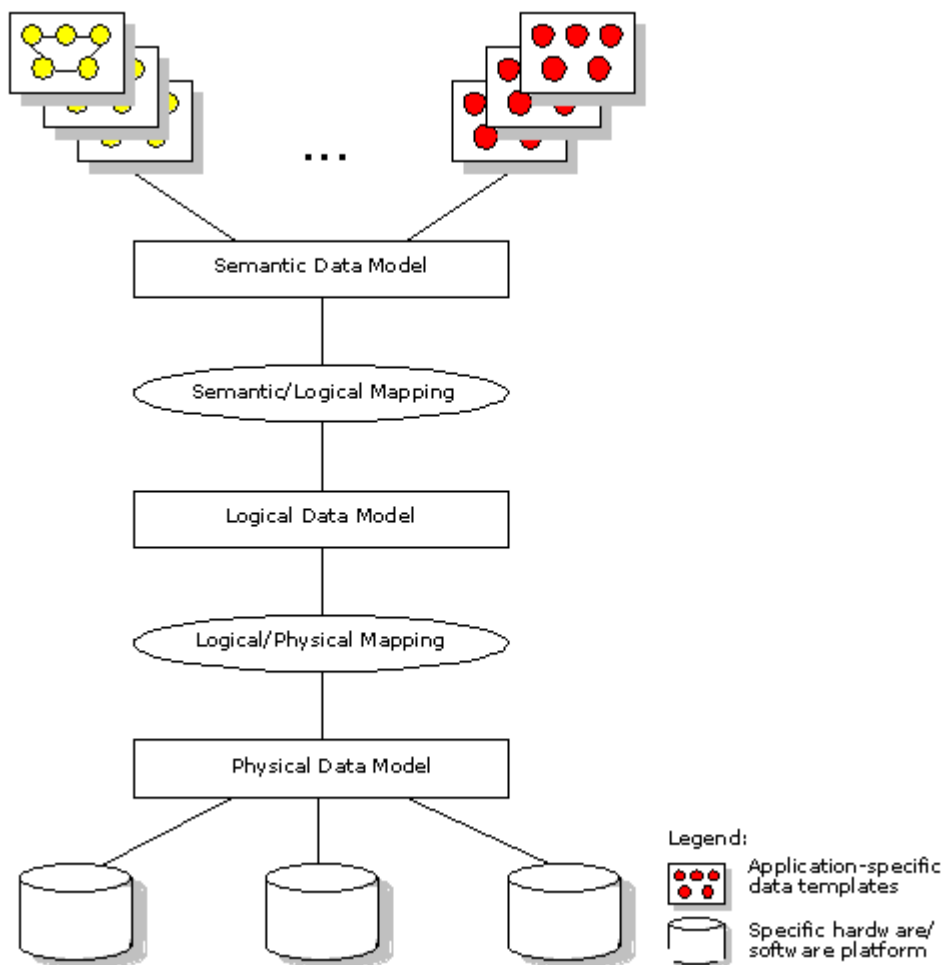


Figure 2.11 ANSI/SPARC database management system architecture

Now, I will apply this well-defined ANSI/SPARC database management system architecture to the Web and analyze the Web from the perspective of the three levels of data modeling as defined by this architecture. Following this standardized approach of identifying the physical, logical and semantic data model of the

Web I will be able to analyze, in a structured manner, all three Web data models. The advantages, disadvantages and problems of data models on three different levels may be easily classified in this way. Such structured approach to the analysis of the Web data model may be seen as the first step of structuring of what is otherwise the most chaotic information system created by mankind.

2.4 Data model of the Web

This section analyzes the Web from the perspective of the above-introduced ANSI/SPARC architecture of database management systems. Thus, the physical, logical and semantic data model of the Web are identified and analyzed in details.

2.4.1 Physical data model of the Web

The Web is a very large repository of HTML and other documents (as already mentioned I reduce here the discussion of other document types on the discussion of HTML documents) residing on a large number of different Web servers. A Web server is accessible via the HTTP protocol. A Web client sends an HTTP request to a Web server. Each HTTP request includes a URL in order to address a particular HTML document residing on the server. Thus, on the physical level, Web servers accept HTTP requests containing a URL of a particular HTML document and reply with this document. That document is visualized on users' screens on the client side.

An in-depth analysis of the physical data model of the Web is out of scope of this paper. Here, we are mainly concerned with the logical and semantic Web data model.

2.4.2 Logical data model of the Web

At the logical data modeling level the Web utilizes the already mentioned basic hypermedia data model, i.e., the node-link data model.

Essentially, the node-link data model supports four basic data manipulation operations: access to a node, addition of a new node, replacement of an existing node and deletion of an existing node. According to these operations we may talk about two sub-models of the node-link data model [Parunak, 1991]:

- Process sub-model
- Data sub-model

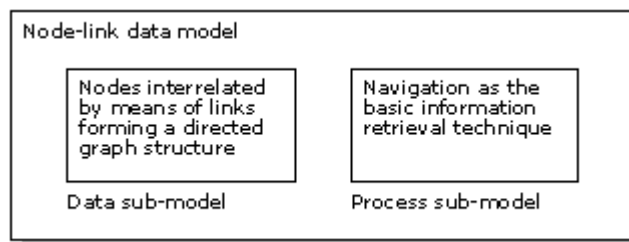


Figure 2.12 Sub-models of the node-link data model

The process sub-model defines information retrieval facilities of the Web, i.e., it is concerned with the issues of browsing the Web. As such, the process sub-model is concerned with the first operation of the node-link data model, i.e., with the operation of accessing a node on the Web.

The data sub-model of the node-link data model comprises the manipulation of the Web and its nodes, i.e., addition, replacement and deletion of nodes.

2.4.3 Process sub-model of the Web

The process sub-model is the first element of the WWW data model. As stated earlier, it comprises the users' operation of accessing nodes on the Web. This sub-model of the Web data model is characterized by its generality and flexibility. Navigation, whereby the user moves through the hypermedia network by activating and following links from one node to another, is a defining feature of hypermedia [Nielsen, 1990]. Another very important property of the process sub-model is its incompleteness. According to the above introduced data model definitions, whereas a data model is a specification of data structures, operations and constraints the process sub-model is an incomplete data model because it does not specify any constraints. For instance, it does not define a detailed specification for navigation access, e.g., how users activate a link, what actions are involved in the process of navigation itself, etc.

Table 2.2 Process Sub-Model

Model	Data Structures	Node operations	Link operations	Constraints
Process Sub-Model	Node and Links	Access	None	None

2.4.4 Data sub-model of the Web

The data sub-model of the node-link data model operates with the basic data structures of the Web: nodes and links. Operations provided by the data sub-model are addition of a node, replacement and deletion of an existing node. It should be especially noted, that the data sub-model does not provide any operations that can be addressed to a link on the Web. All operations are exclusively addressed to nodes.

Nodes are primitive units for organizing information. They may be seen as collections of primitive unstructured data, which are bound together to form a logical entity. For instance, a node may include the textual part, graphics, sound or video clip. Links are other fundamental data units of the data sub-model; they are interrelating different nodes into the Web.

As already mentioned, on the Web links are embedded directly into HTML documents (or in other documents that support hyperlinking), therefore deleting a node will delete also all links that are defined in this particular node. Thus, HTML does not separate the content (the content of an HTML document) from the structure (the navigational structure between different HTML documents), because both structure and content are included in HTML documents. As such, the data sub-model may be seen as a very primitive data model. The separation of structure and content of a database is one of the basic principles of all known database management system and this for many years. However, this separation is not supported by means of the data sub-model. As it is shown in next sections of this chapter the absence of such separation leads to a number of well-known problems of the Web.

The data sub-model is also incomplete. Neither does it define constraints for logical integrity of a resulting hyperweb, nor does it, as already mentioned, define any operations on links. As it will be shown, that fact leads also to a number of well-known problems of the Web.

Table 2.3 Data Sub-Model

Model	Data Structures	Node operations	Link operations	Constraints
Data Sub-Model	Node and Links	Delete Insert Replace	None	None

2.4.5 Semantic data model of the Web

The Web does not support the semantic level of the data modeling architecture at all. The focus of the work presented in this paper lies in defining different approaches for semantic data modeling of the Web structures. These approaches are presented in chapters to follow. However, before the presentation of those semantic data modeling approaches is given an in-depth analysis of the logical data model of the Web is required.

2.5 Advantages of the node-link data model

For many years researchers were analyzing the Web data model. Although this model has a number of advantages over the other hypermedia data models, the list of its disadvantages in the comparison with other models is still bigger. Here I give the overview of the advantages and the disadvantages of the primitive node-link data model. First of all, let us start with advantages of the node-link data model.

The node-link data model is a simple and primitive hypermedia data model. Therefore, this model is:

- Very easy to implement; the both sub-models (data and process) are easy to implement – simple data structures, the small number of operations on the data structures and no constraints at all are reasons for an easy implementation
- General and flexible with the highest degree of freedom; almost everything, i.e., any kind of hyperweb may be created, manipulated and navigated by means of the node-link data model
- Easy to learn for authors as well as for users; information retrieval facilities in point-and-click style are very easy to understand and to use; also, authoring of few nodes and interrelating them by means of computer-navigable links is an easy-to-accomplish task.

Such advantages of the node-link data model were the reasons for the great success of the Web. However, many advanced users experienced, unfortunately, many problems with the Web and its data model. Such problems are, actually, always caused by the shortcomings of this data model. The next section gives an overview of the most known disadvantages and problems of the node-link data model. Such disadvantages of the Web data model were the topic of many researchers' works in the past.

2.6 Disadvantages and problems of the node-link data model

In this section I list shortcomings of the node-link data model. Problems that arise from such shortcomings are described in detail through a number of real-life examples. Further, I provide the overview of some extensions to the node-link data model that were introduced in the past in order to overcome some of such problems. However, there is a number of problems that could not be solved through simple extensions to the node-link data model. To solve such problems more powerful logical data models had to be introduced. The next section provides the overview of the current trends in logical hypermedia data modeling.

2.6.1 Insufficient structuring

The node-link data model is not rich enough (structurally) to support organizing, structuring and accessing tasks required by many applications [Hammond, 1993]. Problems like user disorientation, development of user cognitive overhead and the manual construction of the information network dominate current hypermedia systems [Ramaiah, 1992].

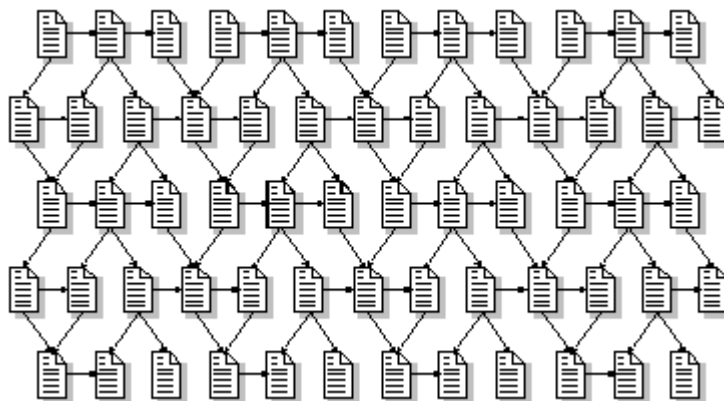


Figure 2.13 "Spaghetti" links

The structure of a hyperweb, created by the means of the node-link data model is a directed graph. However, it is a flat structure residing in a two dimensional plane. Authors have, of course, the freedom to create a hyperweb as they wish, but this freedom leads to a so-called "spaghetti"-like structure of the resulting hyperweb. This can be compared to the "spaghetti"-like structures generated by the use of "GOTO" statement in structured programming languages. As Dijkstra showed [Dijkstra, 1968] such structures are extremely hard to maintain, if the maintaining process is even possible. This problem grows exponentially in the number of nodes and links of a particular hyperweb.

This problem has been identified and discussed extensively in the literature. As Maurer noticed [Maurer et al., 1996]: *as hypermedia systems become larger, more dynamic and possibly distributed, it becomes less and less possible to create and maintain the entire hyperweb of information solely by means of the basic node-link hypermedia paradigm.* De Young suggests that high level structuring mechanisms should be used in hypermedia systems, while pointing out similarities between pointers in programming languages with links and higher level hypermedia structures with abstract data types or class definitions in object-oriented programming languages [De Young, 1990].

2.6.2 Lack of meta-data

In the beginning, there was no possibility in the node-link data model to describe the data (nodes and links) with other data (data about data – meta-data). The lack of the sufficient meta-data constitutes made it almost impossible to find relevant piece of information contained in a node somewhere on the Web.

However, this problem was resolved to some extent by the introduction of the <META> tag [HTML, 2001] in HTML. By using that tag authors were able to provide simple key-value pairs of meta-data describing an HTML document. A number of meta-data initiatives, such as Dublin Core [DCMI, 2001], IMS Metadata Initiative [IMS, 2001], Resource description framework (RDF) [RDF, 2001], etc. were started to overcome this problem.

2.6.3 Logical integrity

As already stated, the data sub-model of the node-link data model is incomplete. It does not define constraints that guarantee the logical integrity of a particular hyperweb. For instance, there is no constraints put on the addition of a new node, which would say how the newly created has to be linked into a hyperweb. It is an exclusively responsibility of an author to integrate the newly created document into the hyperweb. If the author, for instance, forget to do so, the newly added document may not even be accessible from other documents.

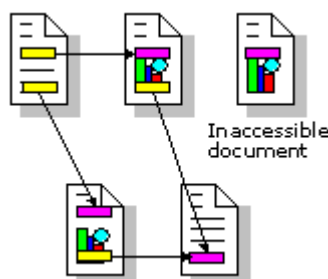


Figure 2.14 Inaccessible document

Another problem is that of so-called dangling links, i.e., links pointing to nowhere or to "nirvana". After a document has been moved or deleted, authors must check all links that point to the deleted document and delete them as well. Otherwise, those links would point to a non-existing document. As Halasz stated [Halasz, 1988]: *the system (based on the node-link data model) cannot reconfigure itself in response to changes to information it contains.*

Referring to these problems Frank Halasz [Halasz, 1991] called this phenomenon the "tyranny of links". He proposed another linking paradigm, where links should not be embedded into nodes but rather they should

be globally addressable objects. Such approach was successfully implemented in a number of hypermedia systems, most notably in HyperWave hypermedia system. In HyperWave, links are the first class objects and are defined by means of a source and a destination anchor. If either of these anchors is deleted the corresponding link ceases to exist. Thus, the referential integrity in this case is guaranteed. The next section describes also other approaches to the solution of this problem, most notably the HM-Data Model [Maurer and Scherbakov, 1996; Maurer et al., 1994; Maurer et al., 1994a; Maurer et al., 1996; Helic et al., 1999].

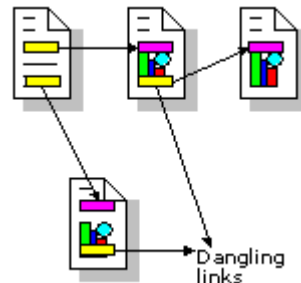


Figure 2.15 Dangling links

2.6.4 Limiting link philosophy

HTML allows authors to embed links into text or graphics. However, links are not just important in text and graphics, but also in video sequences, audio clips and similar.

Recently, a number of different multimedia formats were introduced that allow authors to embed links in animation sequences, streaming video, etc. Such multimedia formats are easily embedded into HTML by means of a Web browser plug-in, such as Flash [Flash, 2001] animations, HM-Card [Maurer and Scherbakov, 1996; HM-Card, 2001] Java applet viewer, etc.

2.6.5 Non-contextual links

In the node-link data model, i.e., in HTML documents residing on the Web, all links are present all the time. When creating a hypermedia document, an author has to make assumptions about the context in which the document will be read. Unlike the text book where readers has just two choices, i.e., to read forwards or to read backwards, in non-linear systems, such as hypermedia systems the boundaries between topics can get rather fuzzy [Halasz, 1988]. Thus, a presence of links, which may be completely unrelated to the current context, rapidly leads to user disorientation.

Let us just consider the following very simple example: consider the “WBT-Master Installation Guide” hyperweb. The “WBT-Master Installation Guide” consists of few HTML documents that form a navigational sequence, i.e., there are always the prior and the next link in each document of the sequence (except the first and the last document which have just the next and the prior link respectively). Suppose now that the “WBT-Master Installation Guide” sequence ends up with the personal homepage of its author say with the document “Scherbakov”. Suppose, that the author of the “WBT-Master Installation Guide” is also a member of a basketball team called “Kings” and that his homepage, i.e., the document “Scherbakov” contains the next link pointing to the next member of the “Kings” team (say “Codd”). Now, the document “Scherbakov” is equipped with the both: the prior and the next link, but they belong to totally different contexts. The prior link points to the last document in the “WBT-Master Installation Guide”, whereas the next document points to the “Codd” document, i.e., to the next player of the “Kings”. However, both of those links are presented to users simultaneously.

Suppose a user starts browsing from the first document in the “WBT-Master Installation Guide” and works through the guide until he/she encounters the “Scherbakov” document. Intuitively, the user may choose to follow the next link in “Scherbakov” that actually leads to the next player of the “Kings” basketball team, namely the “Codd” document. By doing so the user will end up in a context totally unrelated to the starting

one. This leads to total user disorientation, i.e., to a phenomenon well known under the name: getting "lost in hyperspace" syndrome [Bernstein, 1991].

Second generation hypermedia systems try to avoid this problem by introducing a number of very useful navigational tools, such as local maps, site maps, fish-eye views, other hierarchical structures and location feedback [Kappe, 1995].

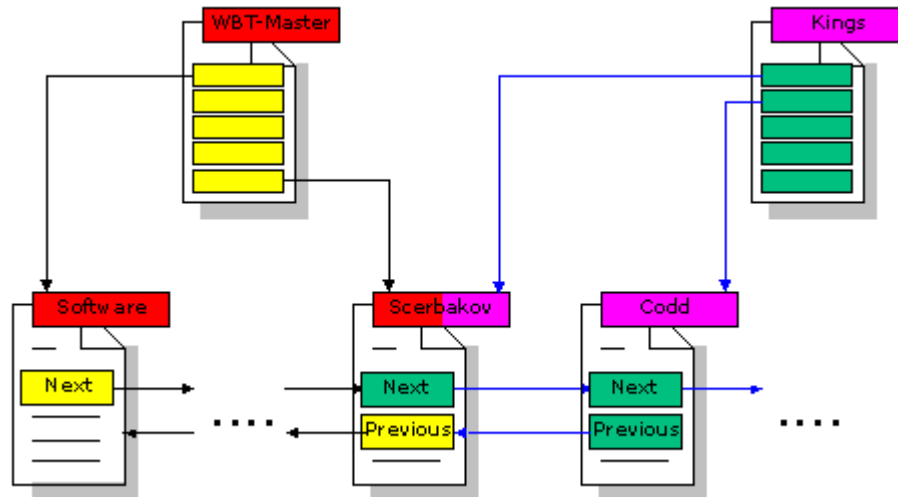


Figure 2.16 Non-contextual links

2.6.6 The "lost in hyperspace" syndrome

Large amounts of information and possible paths to take in hypermedia systems often lead to well-known phenomenon of "user's disorientation" also known as the "getting lost in hyperspace syndrome" [Conklin, 1987; Edwards and Hardmann, 1989]. This problem is due to the incompleteness of the process sub-model of the node-link data model. For instance, the process sub-model does not provide any means to help users locate their current position in a hyperweb.

A number of solutions to this problem is presented in the next section.

2.6.7 Poor search hit rates

The node-link data model and the Web have very poor support for filtered search. Usually, after sending a query users are confronted with the vast amount of totally unrelated information. Additionally, dynamically created pages are still a great challenge to search engines.

The filtered search on the Web was tremendously improved by means of the HTML <META> tags. A number of so-called meta-search engines, such as Google [Google, 2001], Raging [Raging, 2001], etc., gained recently enormous popularity on the Web due to their improved search hit rates.

2.6.8 Inadequate authoring support

This problem may be seen as one of the most important problems of the node-link data problem and the Web. Therefore, let us look on it in more details.

The node-link data model and its data sub-model do not provide authors with facilities for an easy and smooth integration of new documents into the existing system. Recollect that the data sub-model provides just three very simple and primitive operations: addition of a new node, replacement and deletion of an already existing node. Hence, there is no operation that allows authors to directly manipulate the navigational structure of the hypermedia system (e.g., create anchor, create link and similar), but rather they are supposed to carry out this task manually by editing particular nodes. Thus, in the case of the node-link

data model authors are responsible, not only for creating the content of new documents, but also for a proper integration of these new documents in a particular hypermedia system. As Conklin analyzed it [Conklin, 1987]: *Authoring (in hypermedia systems) has much to do with the structuring of ideas, order or presentation and conceptual exploration.*

The basic task of the Web authors may be comprehended as creating the content of a new node and integrating the newly created document into the navigational and conceptual structure of an existing hyperweb. Therefore, the basic authoring task consists of four steps [Maurer et al., 1998]:

- Creating and adding the new node into the hyperweb
- Identifying existing nodes which would be the referring nodes for the new node
- Identifying existing nodes which the new node should refer to
- Setting the links between the identified nodes and the new node.

Evidentially, even the basic authoring task as the Web supports it is far from being trivial one. For instance, in order to identify existing nodes for referring from and to the newly inserted node, authors need to be rather familiar with the hyperweb to the whole extent. In the case of a large hypermedia database, this seems to be rather impossible.

Let us look at the following example, which is, even it is highly simplified and far away from a real situation, an excellent example of how authoring and maintaining link integrity in the Web environment is a rather tedious task.



Figure 2.17 Simple hyperweb

Suppose we have the following hyperweb (see Figure 2.17). The document “WBT-Master Installation” is the index document (shown in the separate HTML frame) and it has links to other documents from the hyperweb (in our case the documents “Required Software”, “Installation”, “Running It” and “TODO List”). The documents “Required Software”, “Installation”, “Running It” and “TODO List” form the navigational sequence, thus the document “Required Software” has a link to the next document in the sequence, i.e., the document “Installation”. The document “Installation”, on the other hand, has a link to the previous document (the document “Required Software”) and a link to the next document from the sequence (the document “Running It”). Also, the document “Running It” has two links: the prior link (the document “Installation”) and the next link (the document “TODO List”) Finally, the document “TODO List” has a link to the previous document from the navigational sequence, i.e., to the document “Running It”.

Suppose that we want to insert the document “Configuration”, which will take the position between the documents “Installation” and “Running It” (see the resulting hyperweb at the Figure 2.19). Obviously, the author is supposed to perform the following operations (as shown in Figure 2.18):

1. Insert "Configuration" into the hyperweb
2. Insert a link into the document "WBT-Master Installation" pointing at the document "Configuration"
3. Update the next link from "Installation" so that it points to "Configuration" instead of "Running It"
4. Set the next link in "Configuration" to point to "Running It"
5. Update the previous link from "Running It" so that it points to "Configuration" instead of "Installation"
6. Set the previous link in "Configuration" to point to "Installation"

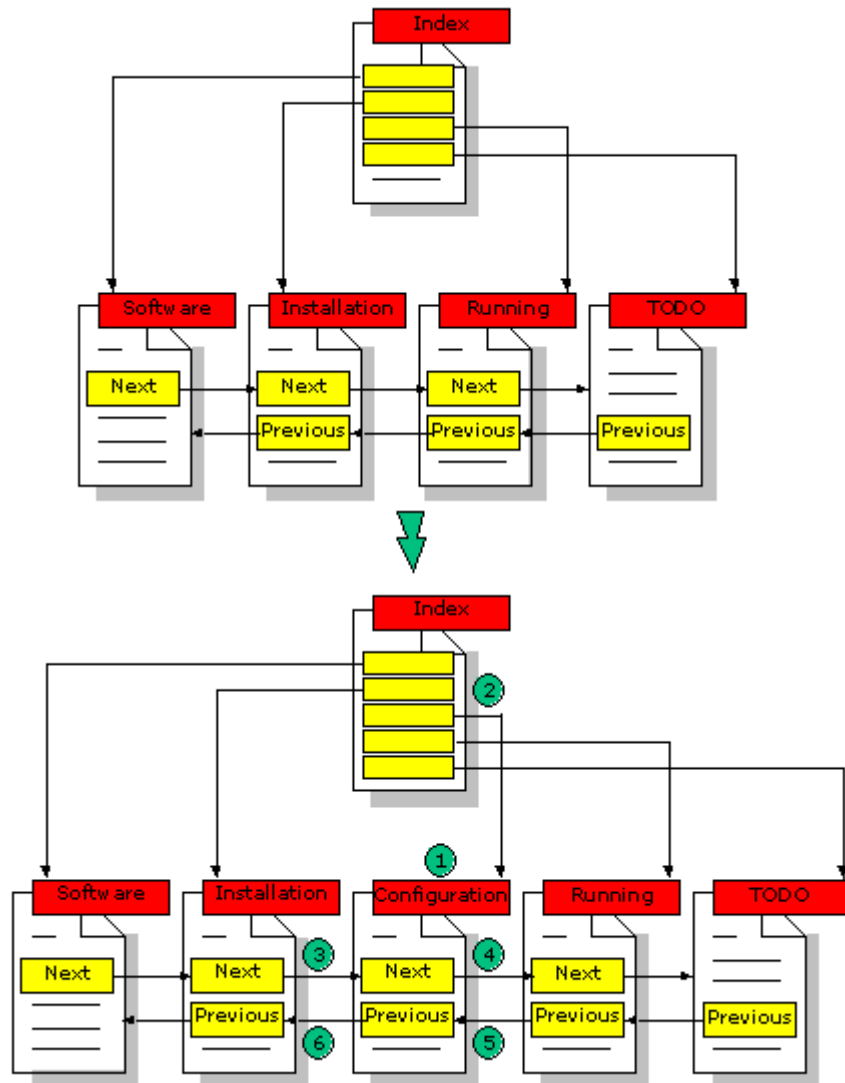


Figure 2.18 Authoring of hyperweb

As it may be seen, just a simple operation of adding a document into the hyperweb with a rather simple navigational structure is, from the authoring point of view, a rather tedious task.

Furthermore, authors are responsible not only for maintaining a particular navigational structure, but also a conceptual structure of the hyperweb. It is well known that the inherent non-linear nature of hypermedia systems tends to create a certain tension between the author and the reader [Bernstein, 1991]. If the author chooses to impose a rigid organizational schema onto the content, it may arouse a degree of resentment. But by offering a richer web of links, the author acknowledges the reader's active participation and bestows upon the reader a measure of liberty and initiative [Hammond, 1993]. Hence, the author has to decide on those issues when creating a particular navigational or a particular conceptual structure of the hyperweb.

Again, a possible solution to the authoring problem is presented in the next section.



Figure 2.19 Resulting hyperweb after modifications

2.6.9 Unsatisfactory reuse of hypermedia material

Reuse - broadly defined as the use of existing information objects or software artifacts in different contexts and for different purposes - is a technology for improving productivity, reducing the production effort and cost, and increasing the quality of end applications, whilst promoting consistency and therefore improving usability.

Reuse is a crucial issue not only in hypermedia applications in particular but in all software applications in general. The reuse of software components in the process of developing new software components has been the main research topic in the software engineering field for many years now.

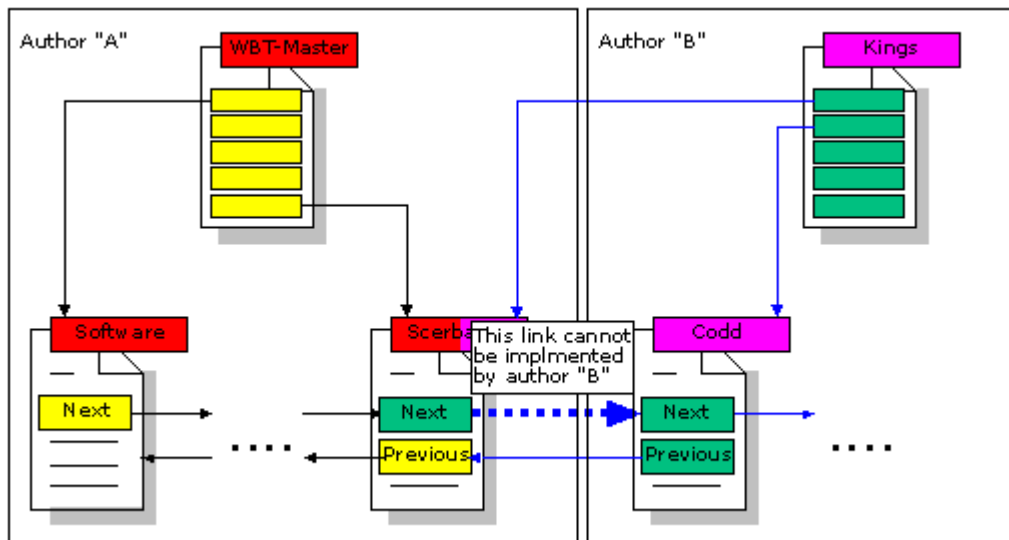


Figure 2.20 Limitations of material reuse

In hypermedia applications, there is no much difference to software engineering field. However, the node-link data model does not provide useful facilities to achieve a satisfactory level of reuse of hypermedia materials. For instance, authors may refer to other documents but cannot embed them locally with possibly new or different hyperlinks [Garzotto et al., 1996]. Given the simple structuring facilities of the node-link data model, this can be done by making an exact copy of the original document, editing this copy to include

different links and inserting the copy into a hyperweb. Nevertheless, such approach is not satisfactory for a wide range of hypermedia authors.

2.6.10 Insufficient support for collaborative work

In the node-link data model there is no means to support structured discussions of number of people, to put annotations to documents, or to work together on a document or collection of documents. More advanced hypermedia data models, such as HyperWave data model provides these features.

2.6.11 Inconvenient security systems

The node-link data model does not provide means to support an encrypted authentication. Because of that charging and billing on the Web is heavily limited. It is a widespread belief that until more convenient, yet secure, costing and billing systems are available, the Web will remain a "toy".

To solve this problem, a number of secure protocols running on the top of the HTTP protocol were introduced, most notably Secure Socket Layer (SSL) [SSL, 2001].

2.6.12 Weak connection to databases

Without tremendous support of a number of workarounds (server-side scripting) there is no possibility offered by the node-link data model to easily connect, manipulate and retrieve information from a database management system. However, recent development in server-side scripting, such as PHP [PHP, 2001] or ASP [ASP, 2001] allows for a better integration of database management systems into the Web.

2.7 Current trends in logical hypermedia data modeling

In the last section a number of well-known problems of the node-link data model were presented. I listed also a number of extensions to the node-link data problem that were introduced in order to overcome these problems. In this section I provide the overview of more powerful logical data modeling paradigms and present the current trends in logical hypermedia data modeling.

As shown, the node-link data model has well-known disadvantages. However, a particular WWW server is not obliged to support the node-link paradigm internally. It can, for example, utilize the relational data model or some other data model to store data. In this case, such "advanced" WWW servers just map dynamically any incoming HTTP request into internal operations and, of course, present resultant data as HTML documents. The mapping mechanism is implemented in a so-called rendering engine running as a front-end application on a remote server [Fernandez et al., 1997; Andrews et al., 1995; Duval et al., 1995].

This approach has been successfully implemented in a number of commercial products (most notably, Home [Duval et al., 1995] and HyperWave).

If we compare all existing proposals for new hypermedia data modeling paradigms with the previously discussed node-link model, we may see the following main trends:

- Extending the basic hypermedia thesaurus containing notions of nodes, links and anchors with new data structures - collections, structured collections, composite components, compounds, containers or whatever. Such new data structures are addressable entities containing other data structures, nodes and links as elements [Maurer and Scherbakov, 1996; Maurer et al., 1994a, Hardmann et al., 1994, Streitz et al., 1992, Garzotto et al., 1991]. I will call such data structures composites in the further discussion.
- Providing some meta-structuring mechanism (often, referred to as templates) to predefine the structure of a number of multimedia documents, thus separating the structure from the content data [Zellweger, 1989].

- Providing multimedia documents with some attributes in addition to the content data [Maurer et al., 1998; Lennon, 1997; Andrews et al., 1995]).
- Upgrading links to independent, addressable objects separated from document content [Maurer et al., 1998; Lennon, 1997; Maurer, 1996].
- Extending the notion of an anchor by associating procedures that locate documents at run-time and even dynamically generate destination documents at run-time [Maurer, 1996].

In the next section I present the HM-Data model, a logical data model, far more powerful than the basic node-link data model that follows some of the mentioned data modeling trends.

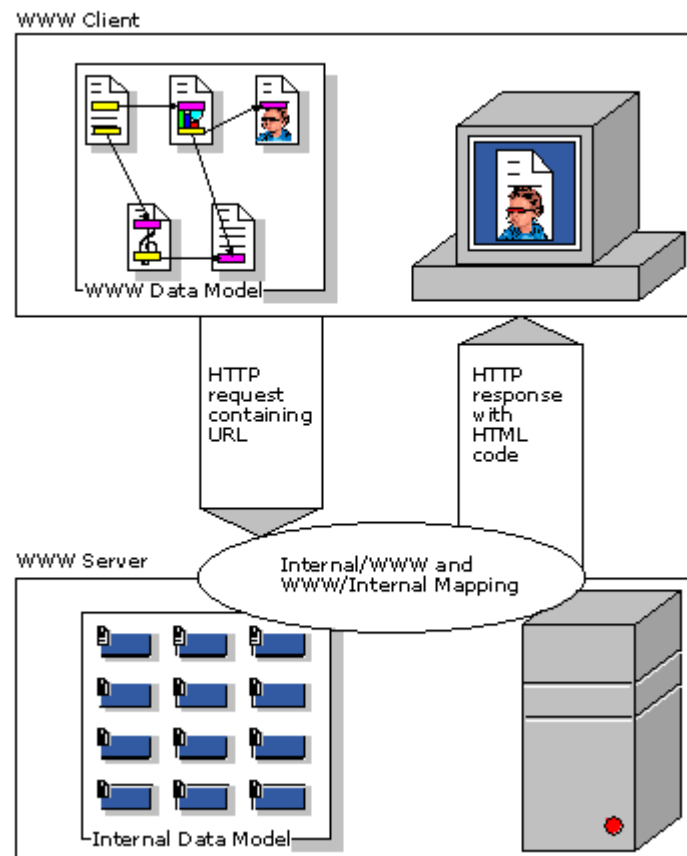


Figure 2.21 Mapping internal data model onto WWW data model

2.8 HM-Data model

The HM-Data Model is a logical hypermedia data model. Hence, it is less concerned with the internals of a document than it is with data structures and operations that can be applied to create, modify and explore (i.e. render) such data structures. Therefore, in the following I will consider multimedia document as atomic, i.e. as basic indivisible chunks of multimedia data that have to be organized. I will examine the use of a novel method of hyper linking for structuring such information chunks into hypermedia databases (hyperweb).

2.8.1 Data structures

A hyperweb, according to the HM-Data Model, consists of addressable composites called Structured Collections (S-collections or just collections, for short). An S-collection does nothing but encapsulates members together with some internal structure (i.e. navigational topology). This structure is in fact a link structure expressing the relationships or associations between members. A member is either a document or another S-collection. One member in an S-collection is chosen and designated by the author as its head.

Additionally, an S-collection may have an associated document called its label to provide contents synopsis. If a label is missing then the head is used instead.

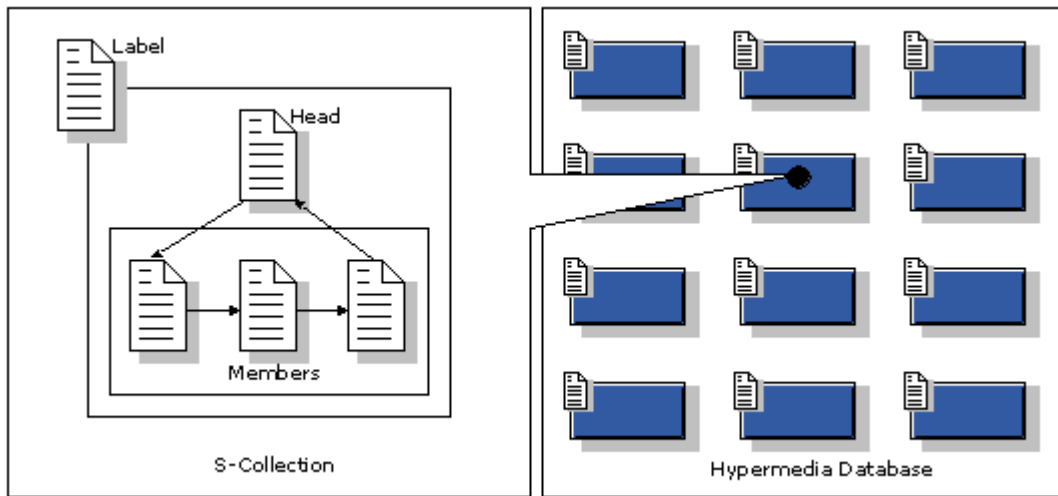


Figure 2.22 Internal structure of S-Collection

2.8.2 Browsing

We may think of an S-collection as an opaque container: if we are outside the container, its members will not be visible to us. To see what's inside it, we must enter it. Of course, we can only be inside one container - the current container - at any given time. But once inside, we will be able to visit its members by navigating its link topology.

A member we visit in the current container may be a document or another S-collection:

- If a document, it will be visualized in some appropriate way – typically involving the presentation of the document's media objects on the computer display and/or sound system.
- If an S-collection, its label (which is a document, by definition) will be visualized. If a label is missing then the head is used instead.

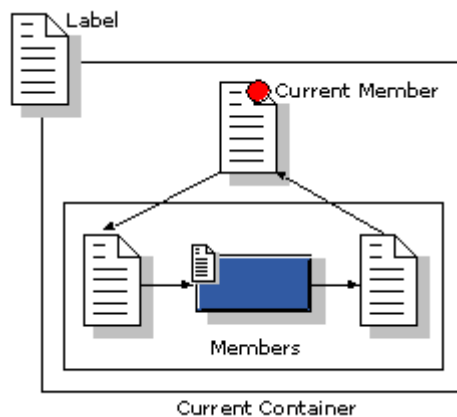


Figure 2.23 Browsing current container

The most recently visited member of the current container is the current member. Access to other members from the current member is determined by the container's link structure: only links emanating from the current member can be selected. All such links are typically visualized as anchors on the current display (these may be icons, hot-words, push-buttons, etc). Navigation within the current container therefore is from member to member as allowed by its link topology.

Note that visiting (i.e. navigating to) an S-collection does not enter it, even though its label is visualized. The point of the label, as mentioned earlier, is to present a synopsis of the collection to allow us to decide whether or not we want to enter it.

To enter it, we must explicitly use the Zoom-In operation. Zooming into an S-collection makes it the new current container, its head the new current member; all links emanating from the current member become accessible.

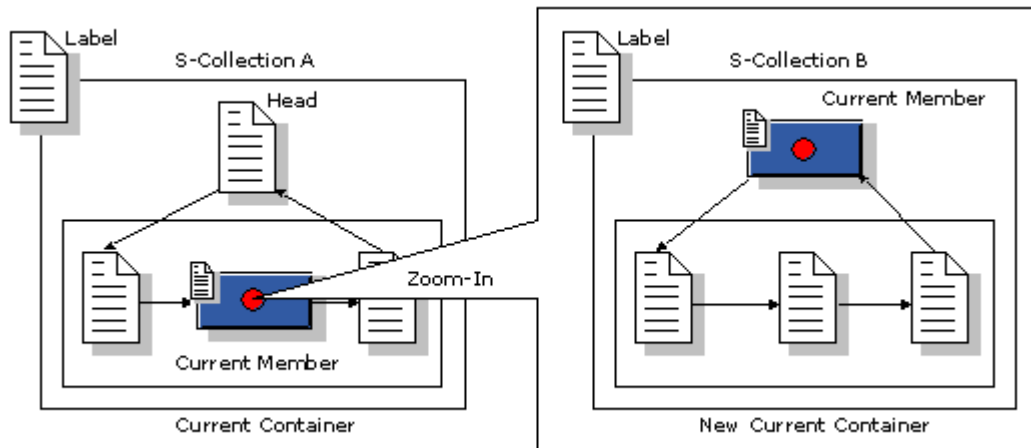


Figure 2.24 Zoom-In operation

Figure 2.24 illustrates the Zoom-In operation.

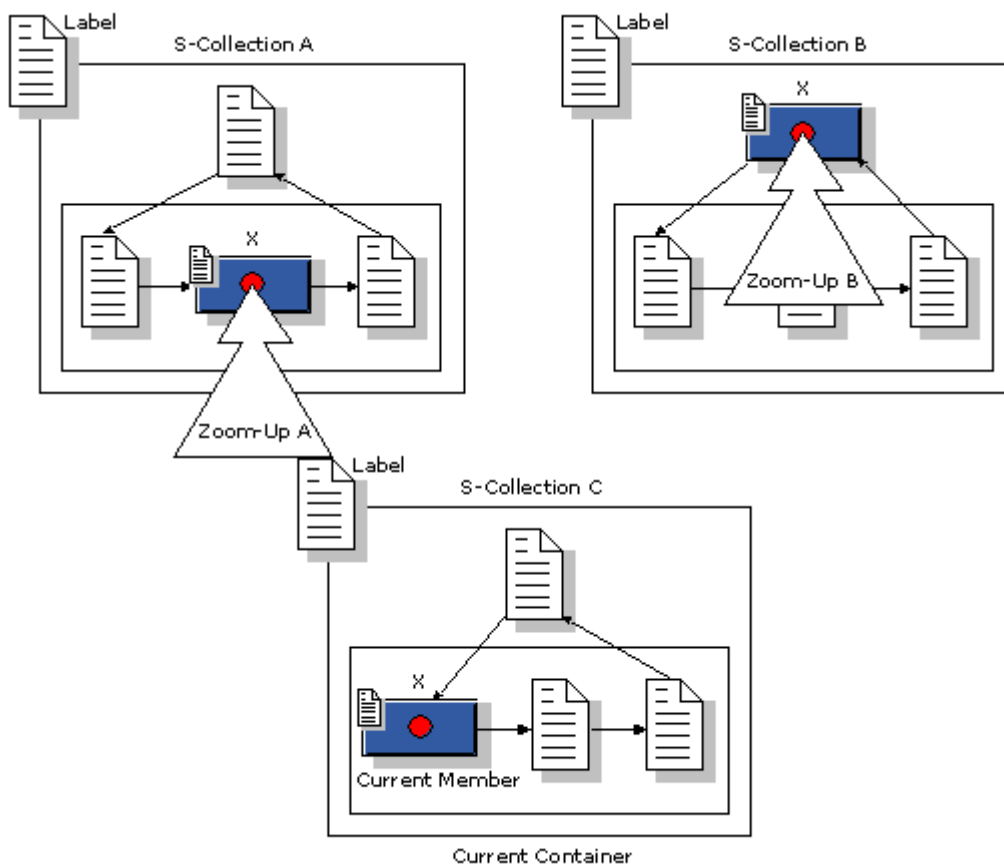


Figure 2.25 Navigational planes

Prior to zooming in, "A" is the current container and "B" the current member. Applying the zoom-in operation on "B" makes it the current container, its head the current member, and all the links emanating from the head accessible. Navigation after zooming in is restricted to the links in the current container, or zooming into the current member (if it is a collection) or zooming out of the current collection.

The Zoom-Out operation is of course the inverse of Zoom-In. Its effect is to restore the current container and the current member to the state immediately prior to the most recent zoom-in. Thus, a zoom-out operation for the situation in Figure 2.24 will reinstate "A" as the current container and "B" as the current member.

Navigation may thus be viewed as occurring in and between "planes". Link navigation within an S-collection is in one plane, while the zoom-in and zoom-out operations are orthogonal to this and affect a jump to a lower or higher plane.

Note also that an S-collection may be a member of more than one other S-collection, i.e. collections may be re-used in different contexts. A zoom-out operation, however, will return to the plane from which it was reached. Thus, if an S-collection "X" (see Figure 2.25) was zoomed in from "B", zooming out gets back to "B" and not "A". This preserves therefore the context of viewing an S-collection. Figure 2.25 illustrates such Navigation "Planes".

One other orthogonal navigation operation is defined in the model: the Zoom-Up operation. This operation affects a jump from the current container to another that also contains the current member. As there may be more than one such container, the operation must also specify which.

2.8.3 Data classes

In the HM-Data Model, a particular S-collection is an instance of one of four predefined types of S-collections:

- Folder
- Envelope
- Menu
- Freelinks.

Each type or data class defines a particular link topology, which constrains the way members can be linked to one another. Additionally, the integrity of the links in any of these types will be automatically maintained.

Figure 2.26 illustrates the link topology of each type:

- Folder connects each member in a bi-directional circular list
- Envelope links each member to every other member
- Menu connects the collection head to every other member using bi-directional links
- Freelinks allows members to be linked in any way desired by the author; the author must ensure in this case that there is a path from the collection head to each member

2.8.4 Creating and modifying a hyperweb in the HM-Data model

Building a hyperweb typically follows a bottom-up approach. First, multimedia documents are created. They then become building blocks for complex S-collections that constitute the hyperweb.

Assuming that documents have been created, an author creates a new S-collection by selecting the desired type (i.e. a folder, envelope, menu or freelinks), assigning it a unique name, selecting a document or an existing S-collection as its head, and optionally selecting a document as its label.

Once an S-collection has been created, members can be inserted, modified or removed. These operations will automatically update the link structure according to the chosen collection type. Insertion of a new member into a menu collection, for example, will automatically create a bi-directional link between it and the collection head. Conversely, removing a member will remove its corresponding link to/from the head. Similar automatic creation and deletion of links apply to envelopes and folders. With folders, of course, we

must also specify the insertion position in the circular list. The exception to all this is the freelinks collection, where the user must explicitly insert and/or remove links to create the desired link topology. Links are second-generation, i.e. separated from rather than embedded in the members. This is of course essential since a member can be re-used in another collection with a different link topology.

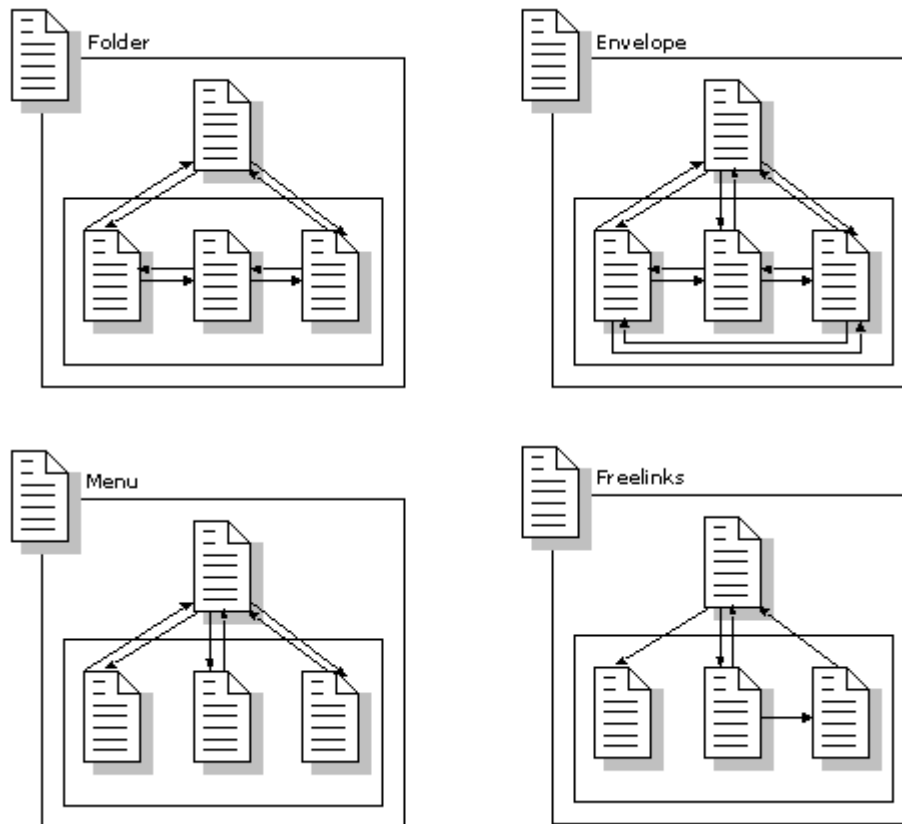


Figure 2.26 Types of S-Collections

2.8.5 Introducing the HM-Data model to the WWW

Formally, a particular type of an S-Collection (i.e. Folder, Menu, etc.) is a meta-definition of a specific linking structure, which is automatically supported by all instances (i.e. S-Collections) of this type. Thus we can say that an S-Collection automatically imposes a particular navigable structure on a top of collections of existing HTML documents or other S-Collections defined as members [Helic et al., 1999].

Practically speaking, we can understand an S-Collection of any type as an entity comprising three elements: label, head and a set of members. Such a data structure may be simply visualized as a special template consisting of three cells [Helic et al., 1999; Helic et al., 1999a].

Similarly, an instance of such data type (i.e. S-Collection) might be seen as a template filled with existing HTML documents and/or other existing S-Collections [Helic et al., 1999a].

Thus, from an author's point of view, there is a number of predefined templates (S-Collection types) where the author can simply insert existing documents or other S-Collections to define sophisticated navigable structures [Helic et al., 1999].

Informally speaking, authors just select previously created documents and/or S-collections from their local file system drag and drop them into other containers (S-collections); links are generated automatically [Helic et al., 1999a].

As stated earlier the Internet Browsers (like Netscape and MS Explorer) of users do not access directly S-Collections. Instead, in order to obtain data, the browser communicates with a special rendering engine, which is a server-side script (if the S-Collection resides on an enhanced WWW server) or a special Java applet (if the S-Collection resides on a local drive) [Helic et al., 1999, Helic et al., 1999a].

In other words, whenever users access such an S-Collection with an ordinary Web browser a special software component is run to visualize the unit in a form of interrelated HTML documents in accordance with the algorithm presented in Section 2.8.2. A so-called CIF file containing a description of the generic link structure and attributes attached to the S-Collection is essentially used to control such visualization. Figure 2.28 shows a possible visualization of the menu authored as shown above.

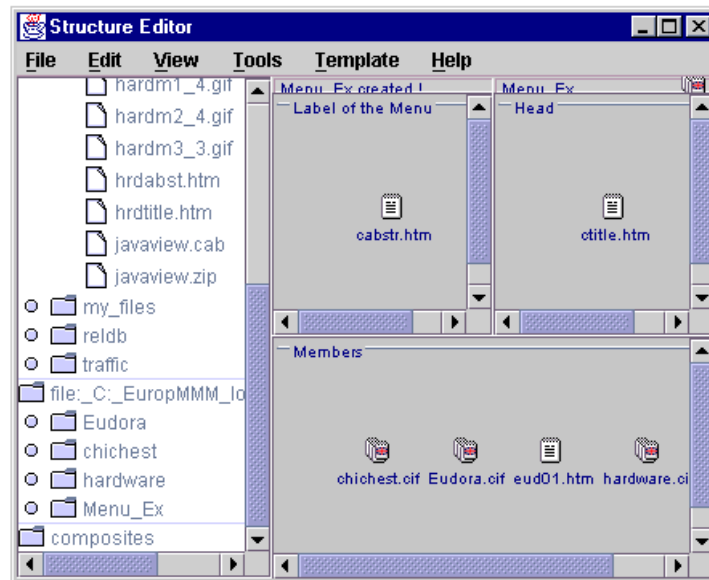


Figure 2.27 Authoring an S-Collection

Links may be visualized in a variety of ways and users, when adding or modifying a member, can typically select how a source anchor is visualized. These may be hot spots, hot-words, scrolling menu lists, icons (as chosen for the example on Fig. 2.29), etc., depending on the particular application of the model.

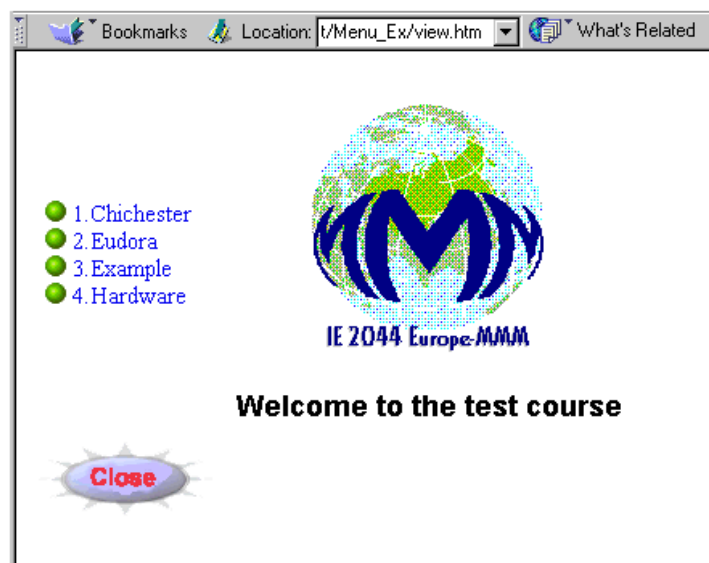


Figure 2.28 Rendering an S-Collection (the menu becomes a current container)

Of course, similar to import/export facilities of many database systems, subsets (of documents and/or collections) of a database may be preprocessed to generate all necessary links as HTML tags, and separately

saved (the option is called "snap" in this particular application). Such a "hardwired" S-Collection can be accessed directly by a Web browser without any additional rendering engine and can hence be used separately. It should be clear, however, that copies are actually made in these cases and that the separated hyperweb is maintained independently, i.e. changes made in source documents and/or S-Collections cannot be seen in this "hardwired" hyperweb [Helic et al., 1999].

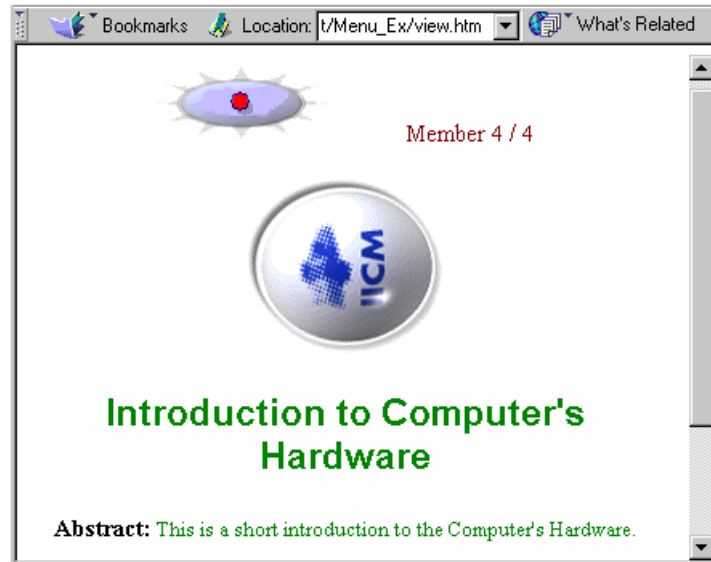


Figure 2.29 Rendering an S-Collection (the member "Hardware" is accessed)

2.9 Discussion of the logical composites approach

2.9.1 Advantages

It should be apparent that the HM-Data Model is highly modular [Helic et al., 1999; Helic et al., 1999a]. Documents and collections can be created independently but at the same time can be re-used through a flexible nesting or containment mechanism. This greatly facilitates multi-author development of hypermedia databases, involving mainly [Andrews et al., 1995; Maurer et al., 1994; Maurer et al., 1994a, Maurer et al., 1996]:

- Creating documents (using some appropriate HTML editor application)
- Creating S-collections of appropriate type (determining therefore their internal link structure)
- Re-using previously created documents and S-collections to define the contents of S-collections being created
- Storing new S-collections into a remote server or local file system (thus making them available for re-use in other S-collections)

Any S-collection can be modified at any time and, as mentioned above, internal link associations will be automatically updated, as members are added/removed. Note, however, that removing a member does not actually delete it from the hyperweb. To delete an S-collection, we must do so explicitly. All collections containing the one being deleted will be updated accordingly.

The HM-Data Model essentially replaces the "spaghetti" view of the basic node-link data model with more structured, independent but fully compatible hypermedia modules called S-Collections.

In summary, the following features of the HM-Data Model distinguish it from other existing logical hypermedia data models [Helic et al., 1999]:

- Links neither belong to individual nodes nor are they globally addressable objects. Instead, they are encapsulated in hypermedia containers called S-collections. By definition, links exist only between members of a collection, i.e. links cannot be created to destination nodes that are outside the

collection. S-collections therefore represent well-defined chunks of information that may be re-used in different contexts without concern for superfluous links.

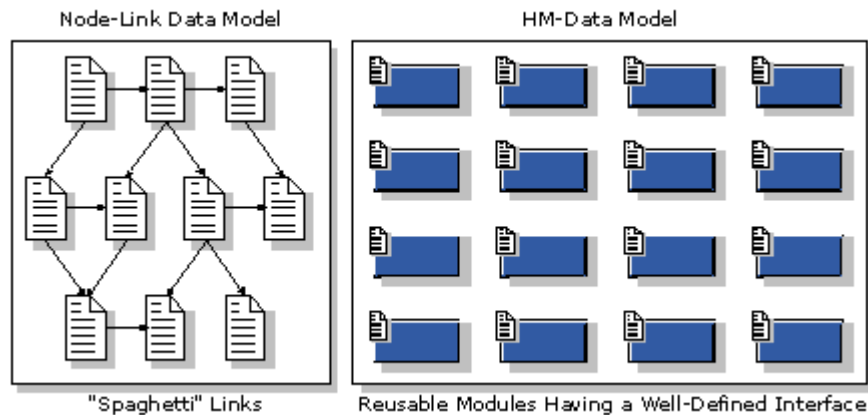


Figure 2.30 HM-Data Model vs. Node-Link Data Model

- The restriction of links to local contexts only is compensated by orthogonal navigation operations of Zoom-In, Zoom-Out and Zoom-Up.
- Containment of complex collection objects within another, referred to as "re-use" rather than "reference", is in line with object-oriented views and closer to the intended concept of sharing existing resources, particular in different contexts. The term "reference" too strongly suggests "jumping to another location" (with a fixed given context). "Re-use" on the other hand implies a (logical) embedding of the external object into the current context (switching between embedding contexts is via Zoom-Up).
- Authoring is effected by member wise inclusion of hypermedia chunks, as opposed to "spaghetti" linking of nodes in other models. Any S-collection can be included in any other. Recursive membership relations remain a possibility and will allow the modeling of arbitrarily complex hypermedia databases.
- All operations are addressed to a particular data object (S-collection) and do not affect the link structure of other objects. This object-oriented character of the model presents new ways of supporting the logical integrity of hypermedia databases.

2.9.2 Disadvantages

Even though the application of logical composites to the WWW has a large number of advantages there is still a number of shortcomings of such an approach [Helic et al., 1999b]. Actually, those disadvantages may be applied to any other logical hypermedia data model, not only to the HM-Data Model:

- Authors and users, that are used to work just with the node-link data model as supported by the WWW, have to learn how to work with a new data model. Such a learning of new data structures, operations and constraints of a new logical data model may be a rather time-consuming task.
- Data structures introduced by logical data models are generic data structures, i.e., they are to be applied in any application, regardless the application-specific issues. For instance, let us consider a WBT application. In such applications it is much more comprehensive for both authors and users to think in terms of say, courses and chapters of courses, than in terms of folders, menus or envelopes. In a WBT application, a course, a learning unit or a chapter would be a data structure of choice for authors and users of such applications.
- Similarly to data structures, operations in logical data models are generic and one and the same in all applications. However, we may easily imagine that in a WBT application an operation that shows a course map would be quite useful.
- Actually, the two last items stated an absence of a possibility to introduce the application-specific data structures and operations. In the database theory a Data-definition-language (DDL) is used to define data structures, operations that can be applied to the data structures and constraints that have to be satisfied. As we may see a typical logical hypermedia data model does not provide means of a DDL, which could be applied in order to introduce purpose-oriented data structures and operations.

If we would have such a possibility, we could define new data structures that would be a better fit (than generic logical data structures) for a particular application we had in mind.

2.10 Limitations of logical hypermedia data modeling

The concept of logical hypermedia composites is an important logical hypermedia data-modeling concept.

As already stated, this concept has been applied in a wide range of different hypermedia applications with a great success. Among those applications is HM-Card, as well as Hyperwave Information Server, to mention just a few. Furthermore, a lot of research projects were done regarding hypermedia systems utilizing the concept of hypermedia composites.

Thus, this concept may be seen as a kind of a reference logical hypermedia data-modeling concept. Therefore, the discussion of limitations of such an approach may be easily applied to any other logical data modeling approach.

As the discussion of the HM-Data Model (a typical logical hypermedia data model utilizing hypermedia composites) showed, hypermedia composites provide a useful data-modeling paradigm, which solves a wide range of problems of the primitive node-link data model.

Even though, this approach is much more powerful than the primitive node-link data-modeling paradigm it has also a number of shortcomings. The above discussion listed a few of them. Those shortcomings are potential cause for new problems and limitations in hypermedia systems utilizing hypermedia composites.

As I look on the hypermedia composites as a kind of a reference logical hypermedia-modeling paradigm I conclude that similar disadvantages (problems, limitations) may appear in any logical hypermedia data model, i.e., these disadvantages are inherent to logical data modeling as such.

I claim that those limitations of logical hypermedia data modeling approaches may be resolved only by applying semantic hypermedia data modeling concepts. Let us take a look on the following examples of hypermedia applications [Helic et al., 2001; Helic et al., 2001a; Helic et al., 2001b]:

- A logical hypermedia composite defines a particular navigational structure on the top of its members. Members of a composite are defined as a label, a head or a normal member. However, sometimes authors want to define different roles for members within a composite, such as a title, an abstract, chapter of a composite [Helic et al., 1999b]. Also authors may want to apply a different navigational and/or visualization paradigm on the top of members of a composite. Obviously, there is a need for a tool to define new types of composites, with different navigational structures and different members roles. As already mentioned, in traditional database management systems such tool is called Data-definition-language (DDL), allowing database administrators to define new data structures, their properties and their behavior. These new types data structures are used to instantiate a number of instances that inherit one and the same properties and behavior. The process of defining and instantiating new composite types may be also seen as a process of *knowledge structuring*. Obviously, such process is happening on the level of Meta structuring (semantic hypermedia data modeling) and is not supported by means of logical data modeling concepts.
- A navigational structure provided by means of logical hypermedia composites is not always sufficient to provide users of hypermedia systems with a general overview of a hypermedia database [Helic et al., 2001a; Helic et al., 2001b]. Such a navigational structure reflects a way of accessing and working through a particular hyperweb by users of hypermedia systems. Such a "navigational oriented" data structure consisting of documents "B" and "C" mainly prescribes reading "B" before reading "C" and has nothing to do with a possible situation that "C" may be a documentation on a software module implemented by the programmer "A" for a project "B". Often, users need a general overview and access to all documents provided by a particular hypermedia system. Speaking in the same primitive language, as before, such overview should provide users with a profiled knowledge about documents "A", "B" and "C" saying that "A" is a programmer, "B" is project and "C" is a

software module. Thus, there is a need in hypermedia systems for tools that support the process of *knowledge profiling*. Again, such process may not be supported by means of logical hypermedia data models because it involves meta-structuring mechanisms.

- Often, users of hypermedia systems are confronted with a large number of different information chunks (HTML documents, structured hypermedia composites, etc.). Finding a relevant information chunk constitutes a rather difficult problem. Thus, users are involved in a tedious task of *knowledge mining* trying to find and access the best-match hypermedia resources [Helic et al., 2001]. There is a number of navigational tools that help users during such process. If users would be equipped with a possibility to browse not individual documents or hypermedia composites, but rather concepts represented by those resources the process of knowledge mining would be much easier. However, that includes again a possibility to structure a hypermedia database on a meta-level by identifying concepts and assigning resources to such concepts. Again, logical hypermedia data modeling concepts are not sufficient.

Let us now look closely on the above mentioned *knowledge transfer* processes in hypermedia systems through a number of examples.

2.10.1 Knowledge structuring

Let us consider the following example. Suppose an organization maintains a Web-based-training system. An experienced employee (tutor) is supposed to conduct distance-training sessions for their employees on regular basis. The tutor in collaboration with a courseware author develops a special Internet course and makes a special announcement on the WBT server. Potential learners may access the announcement board and subscribe themselves for a particular training session.

Now, let us closely examine the process of authoring of a number of special Internet courses. Obviously, this is a process of authoring a hypermedia database, i.e., it consists of the following steps:

- Preparing a number of hypermedia documents (a course title, a course abstract, different chapters of a course)
- Interrelating the prepared documents into a navigable structure (chapters, courses)
- Mapping the prepared navigable structure onto the Web.

Suppose, that the WBT server allows authors to create and maintain hypermedia composites. Thus, the author creates a number of composites and combines them with the prepared documents into a resultant composite representing the course. Each of those composites provides a particular navigational structure (sequence, menu, etc.) and defines different member roles (label, head, etc.).

Obviously, the author's choice of a navigational structure and/or roles of members within a composite is limited by the number of different composite types. The author has no possibility to introduce different roles for different composite members or to implement another navigational and/or visualization paradigm for a composite.

For instance, the author may want to introduce a course data structure, which consists of a title document, an abstract document and a number of composite chapters. Chapters in turn may have a title document and a number of pages. The author may have the following navigational and visualization paradigm in mind:

- Whenever a course is accessed the title and the abstract document are shown combined into an HTML document together with a list of links to each particular chapter
- A chapter is visualized with links to the starting course document together with links to the previous/next chapter in the chapters' sequence.

However, logical hypermedia data modeling paradigms allow authors to create generic data structures, which are one and the same regardless, a particular hypermedia application. There is no possibility to introduce application-specific or purpose-oriented data structures.

Obviously, meta-structuring mechanisms are needed here. For instance, a Data-definition-language that allows defining new composite types with possible different member roles and possible different navigational and/or visualization paradigms could solve such a problem [Helic et al., 1999b].

The process of structuring the problem domain at a meta-level, i.e., the process of identifying new composite types and different member roles for those types is a modeling process that includes representing the knowledge existent in the problem domain. I call this process knowledge structuring in a hypermedia database.

On the other hand, users confronted with such semantic hypermedia composites can much more easily comprehend information contained in it.

Thus, the process of knowledge structuring consists of three processes:

- Defining new types of hypermedia composites with possible different member roles and navigational structure
- Creating instances of different composite types and assigning existing hypermedia resources to different member roles
- Accessing and browsing instances of different hypermedia composites.

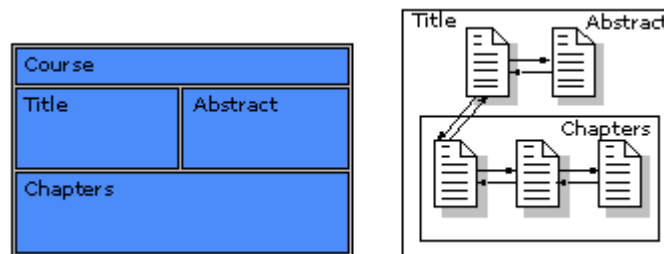


Figure 2.31 Knowledge structuring: Defining new composite types

2.10.2 Knowledge profiling

Now, let us consider the following example. Suppose a company needs to install knowledge profiles for each of its employment positions in a hypermedia database. The process may be seen as defining a position accompanied with descriptions of knowledge components required for this position. Moreover, the knowledge components may be further associated with hypermedia resources which provide such knowledge. On joining the company an employee needs to be inducted into the practices and procedures of the organization. The new employee undertaking the associated training for the prescribed knowledge required for the position they hold will facilitate this. New or trainee employees may always see what knowledge is needed (i.e., they can browse the position profile) and automatically access the training resources if any additional training is required.

One obvious solution to this problem is to use a standard hypermedia-modeling paradigm, i.e., the node-link data model. Thus, persons responsible for creating knowledge profiles would create a number of HTML pages and interrelate them into a hyperweb. However, considering a big organization with a vast number of different positions this can facilitate a rather difficult task. Moreover, maintaining such a hyperweb would be a rather tedious job. For instance, introducing a new position and linking it into already existing hyperweb could become a nightmare for authors of that hypermedia database.

Let us consider another possible solution. First, let us try to identify a possible semantic entities (categories, concepts) involved in this particular application. Obviously, we have the following three different entities:

- A position
- Knowledge required for a position
- Hypermedia resources that contain the required knowledge.

Second, let us try to identify semantic relationships that relate those semantic entities (we can read the relationships from the entities). This, we have:

- Required for: relates knowledge and position
- Contains: relates resource and knowledge.

Now, if we apply the identified semantic categories and their relationships to the authoring process of that hypermedia database the authoring process would be very simplified. For instance, authors would be able to use the defined schema and just assign different hypermedia resources to the defined semantic categories. Further, if the process of relating those resources into a navigable structure by means of semantic relationships is automatic, the authoring process is rather simple task to accomplish.

Again, the process presented here may be seen as the process of structuring a hypermedia database on a meta-level, i.e., identifying different semantic categories and interrelating them by means of semantic relationships. We call this process knowledge profiling [Helic et al., 2001b].

Also, users access instances of semantic categories and browse links that are much more understandable to them.

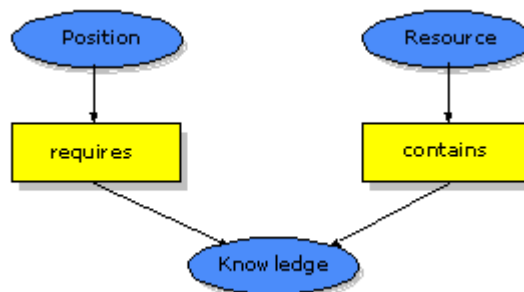


Figure 2.32 Knowledge profiling: Identifying categories and relationships

Thus, the knowledge profiling process consists of the following processes:

- Identifying semantic entities and interrelating them by means of semantic relationships at a meta-level
- Creating and maintaining instances of different semantic entities in a hypermedia database
- Accessing and browsing instances of semantic categories.

2.10.3 Knowledge mining

Finally, let us consider the following example. Again, an organization is maintaining a large hypermedia database in the form of a Web Based Training system. A learner needs training on a particular subject to acquire additional knowledge, and is aware about that WBT server containing relevant information. The learner accesses the server to find most relevant training material, to work through these materials and to communicate with the subject experts and with other learners working on similar materials.

Now, the WBT server contains a large number of hypermedia (or any other) resources. For instance, there could be a number of Internet courses, structured according to hypermedia composites approach. Also, the WBT server may contain a large number of HTML pages, WinWord or any other kind of documents. Further, different discussion forums residing on the server are also valuable learning resources.

Confronted with such vast amount of hypermedia information users are often confused and not able to find and access the best-match hypermedia resource. Obviously, allowing users to navigate through the concepts that particular hypermedia resources describe, rather than through the hypermedia resources their self could resolve easily the users' confusion.

Again, identifying different concepts and assigning hypermedia resources to those concepts is a meta-structuring process. A hypermedia database equipped with such a conceptual map of hypermedia resources

allows users to find and access instantly relevant hypermedia resources. Thus, it reduces considerably users' effort put in the process of what we call knowledge mining [Helic et al., 2001].

Thus, the knowledge mining process consists of the following processes:

- Identifying important concepts in a hypermedia database
- Assigning hypermedia resources to those concepts
- Accessing best-match hypermedia resources by navigating the conceptual map.

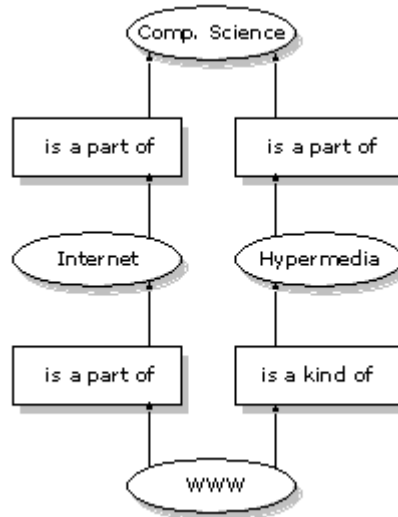


Figure 2.33 Knowledge mining: Identifying concepts in a hypermedia database

3 Semantic Hypermedia Data Modeling

The last chapter introduced the important logical data modeling concepts in hypermedia systems in general and the WWW in particular. The chapter was concluded with a list of limitations of logical hypermedia data modeling. Introducing semantic data modeling approaches might solve those limitations.

This chapter presents solutions to the limitations of logical hypermedia data modeling.

3.1 Semantic data modeling in general

In its most general sense a semantic data model defines a number of semantic entities (categories, concepts) interrelated by means of semantic relationships (relations between such entities), which model a particular subject domain [Rishe, 1992; Borgida, 1991; Peckham and Maryanski, 1988; Hull and King, 1987; Minsky, 1968].

Actually, such model provides a conceptualization of that particular subject domain. Semantic data modeling is similar [Rishe, 1992; Borgida, 1991; Peckham and Maryanski, 1988; Minsky, 1968] to different knowledge representation techniques [Brodie et al., 1984; Calvanese et al., 1998; Dillon and Tan, 1993; Minsky, 1975]. Both of these modeling approaches utilize conceptual modeling [Sowa, 1984; Borgida, 1987; Brodie et al., 1984; Minsky, 1968] of a particular domain. However, there are some differences [Borgida, 1991] as well. Whilst knowledge representation might be understood as a *description* of a particular data set, semantic data models are *prescriptions* of how a particular data set should be modeled. However, mechanisms, methods and representation techniques in both these approaches are quite similar. Thus, I will apply such methods in both senses, i.e., prescriptive (for instance for defining a database schema) and descriptive (for instance for identifying concepts and relationships in a particular data set).

Usually, in formal approaches [Sowa, 1984; Nosek and Roth, 1990] to semantic modeling a set of symbols (an alphabet) with well-defined syntax and semantic is used to describe concepts of a particular semantic data model. A data model (interpretation) is an actual assignment of data to defined entities and their relationships. There exist also less formal (semi-formal, informal) semantic data models.

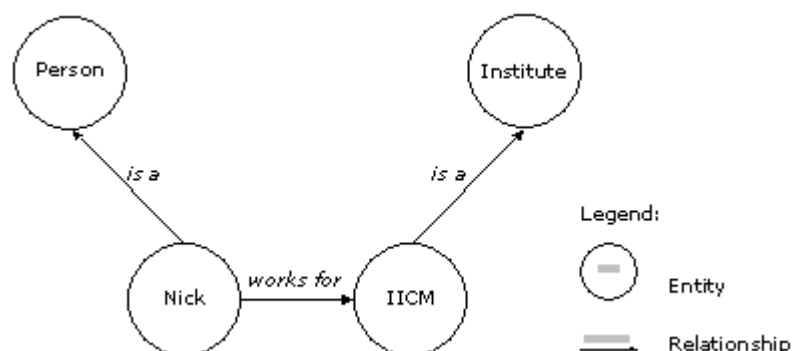


Figure 3.1 Entities and relationships

There are several different kinds of representation [Sowa, 1984; Lehman, 1992; Calvanese et al., 1998; Minsky, 1975] of semantic data models [Al-Khatib et al., 1999; Rische, 1992; Borgida, 1991; Peckham and Maryanski, 1988; Hull and King, 1987; Nosek and Roth, 1990] (according to the art of the alphabet that is used):

- Logics representation
- Network representation
- Object representation.

Logics representations [Calvanese et al., 1998; Nosek and Roth, 1990] use well-known mathematical apparatus of the traditional logics. Usually, predicate logics (first-order or higher-order predicate logic) is used to model a particular subject domain. The field of the traditional logics has been investigated for many years, thus allowing for creating of highly sophisticated semantic data models.

Network representations [Calvanese et al., 1998; Nosek and Roth, 1990] allow for creating of graphical representations of data models. Usually, such representations provide useful graphical overviews of a particular subject domain, where users can see at a glance, concepts and their interrelations inherent to that subject domain. Thus, such representations are very comprehensive and extremely understandable. Among such representations are semantic networks [Sowa, 1984; Lehman, 1992; Borgida, 1987; Nosek and Roth, 1990; Sowa, 1991], directed labeled graphs [Sowa, 1984], conceptual graphs [Sowa, 1984], etc.

Finally, object representations [Calvanese et al., 1998; Dillon and Tan, 1993] allow for representing of a particular subject domain in the terms of objects, their types, properties, relations to other objects/types, and their behavior. These representations provide similar mechanisms as object-oriented programming languages. Usually, such representations support a number of important modeling concepts such as [Dillon and Tan, 1993]:

- Generalization [Sowa, 1984; Borgida, 1987] of an object type(s) to a more general type. For instance, object types “Article” and “Book” might be generalized to an object type “Document”. In object-oriented programming languages “Document” might be a super-class of “Article” and “Book” class.
- Specialization (classification) [Sowa, 1984; Borgida, 1987] of an object type to a more specific type. This concept is complementary to that of generalization, i.e., the object type “Document” might be specialized to object types “Article” or “Book”. Again, in object-oriented programming languages we have a pendant to this concept: sub-classing.
- Aggregation [Sowa, 1984; Borgida, 1987] of two or more object types to form a new compound object type. For instance, a number of “Article” objects might be aggregated to a “Book” object. Again, in object-oriented programming languages we have a possibility to compose a class from already existing classes.
- Instantiating of objects (instances) [Sowa, 1984; Dillon and Tan, 1993] of a particular object type. For instance, we might create instances of the “Article” object type with names “Article 1”, “Article 2”, etc. All object-oriented programming languages provide means for creating class instances at the run-time. Usually, there is a special keyword (new) supporting this concept.
- Inheritance [Sowa, 1984; Dillon and Tan, 1993] allows instances of a particular object type to inherit all properties and behavior of that type. For instance, if the “Article” object type is defined to represent short documents with maximal 5 pages then all instances of this type, such as “Article 1”, “Article 2”, etc., inherit these properties, i.e., they are all short documents with maximal 5 pages. In object-oriented languages this concept is one of the fundamental concepts. It allows instances of different classes to inherit properties and behavior of their class, thus providing all instances of a particular class with a common programming interface.
- Defining domain taxonomy [Sowa, 1984; Dillon and Tan, 1993]. Taxonomy is usually defined as object class hierarchy. Such definition is very useful as it provides a hierarchical overview of objects and their classes, which are part of a particular subject domain.
- Defining domain ontology [Staab and Maedche, 2001; Luke et al., 1997; Simons, 1987; Gruber, 1993; Gruber, 1993a]. Ontology is defined as specification of a conceptualization, i.e., it defined important concepts and their relationships. This modeling concept is very important because it provides a mechanism to separate a so-called database schema (i.e., definition of a database

structure) form the actual content of a database (i.e., assignment of database items to concepts defined by particular domain ontology).

Usually, semantic data models provide a higher level of abstraction of a particular subject domain than logical data models. It allows for describing of the subject domain in the terms that are inherent to this subject domain. On the other hand, logical data models describe a particular subject domain in the terms of a number of predefined generic data structures.

Again, we can draw a parallel with programming languages. For instance, an assembler language represented an abstraction of the underlying machine, thus allowing programmers to program by means of simple instructions (add, multiply, divide, etc.) rather than by means of sequences of bits.

Higher procedural programming languages provided an abstraction of the assembly language. Now it was possible to program in terms of structural programming blocks, functions, procedures, etc. However, when solving a particular programming task programmers still had to think about the problem in the terms of the solution domain (i.e., in the terms of the underlying machine).

For the first time, with the introduction of object-oriented programming languages programmers got a tool where it was possible to think about the problem in the terms of the problem domain (for the most of the time), i.e., to solve the problem on a computer by programming a number of classes, which represent the problem on a level of abstraction that is much closer to the real-life situation.

Similarly, semantic data models provide such abstraction in order to describe data sets on the level much closer to the real-life than it is possible with a standard logical data model.

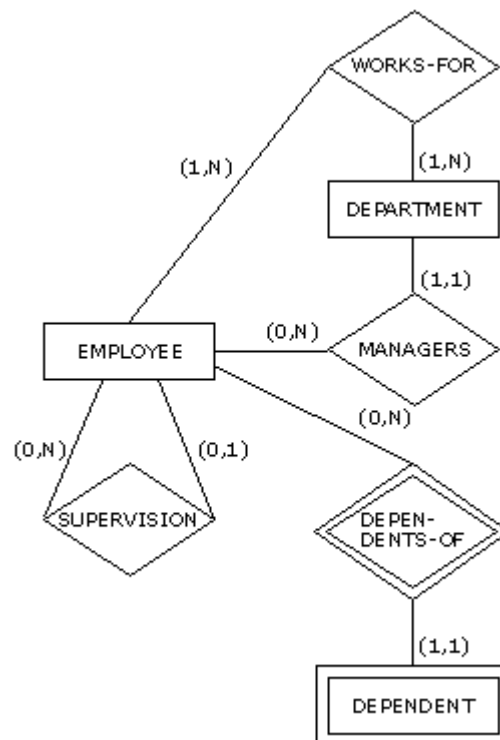


Figure 3.2 Typical ER-diagram

Usually, data sets described in the terms of semantic data models are much easier comprehended by humans and are much more made interoperable between different computer applications. For instance, in a typical Web-based-training (WBT) application the basic data structure may be a “Course”. A “Course” may consist of a “Title”, an “Abstract” and a number of “Chapters”. Either for users who navigate within a “Course”, and especially for authors who create “Courses” is much easier to comprehend and successfully use a WBT system that supports data structures such as “Course”, “Title”, “Abstract” or “Chapter” than a WBT system

that supports “Courses” as S-Collections with a label, a head and a number of members. Internally, such WBT system may support HM-Data Model but the semantic level of data modeling with semantic/logical and logical/semantic mapping mechanisms should be present in the system as well. Moreover, if we consider another WBT system where a “Course” has a “Title” and a number of “Pages” it is trivial to make the courses from these two systems interoperable. A simple mapping definition (mapping “Chapters” from the first “Course” onto “Pages” from the second one and mapping “Abstract” from the first onto “Page” number 1 in the second one) is all that is required to make these two systems compatible with each other.

Let us now look on an example of a semantic data model applied in traditional database management systems: Entity-relationship (ER) model [Hull and King, 1987]. It may be seen as an abstraction of the well-known relational data model to the terms of entities and relationships. Mostly, entities and relationships from an ER diagram are directly mapped into relations of a relational database schema.

3.2 HC-Data model: Knowledge structuring in hypermedia systems

As described above in the section 2.10.1, the process of knowledge structuring in hypermedia systems cannot be supported by means of logical hypermedia data modeling concepts. Rather, semantic hypermedia data modeling must be applied. I developed a semantic hypermedia data model, called the HC-Data Model that provides means of knowledge structuring concepts in hypermedia systems.

The Entity-relationship model was applied as an abstraction of the relational data model to provide a more powerful knowledge structuring mechanism traditional database management systems.

In the same way, the HC-Data model [Helic et al., 1999b; Helic et al., 1999a] is an abstraction of the HM-Data model. The HC-Data model speaks just in terms of hypermedia composites and their members. Roles of members in a particular hypermedia composite may differ from one composite type to another, as well as the navigational structure of a composite. A special Data-definition-language supported by the HC-Data model may define those roles and the navigational structure [Helic et al., 1999b]. DDL provides facilities to define new types of hypermedia composites, which may be used to instantiate a number of instances sharing common properties of this particular type. In the HC-Data model the type definition is comprised as an HC-Type, whereas instances of a particular HC-Type are called HC-Units. The HC-Data model inherits the basic browsing operations such as Access, Zoom-In, Zoom-Out and Zoom-Up, as well as integrity constraints from the HM-Data model.

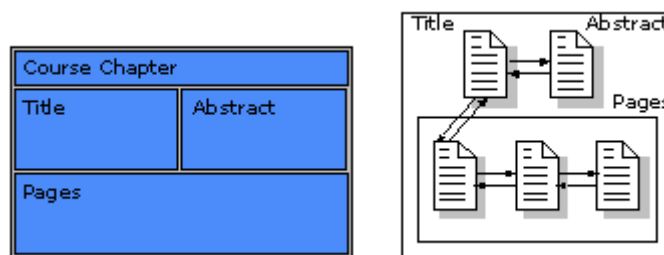


Figure 3.3 HC-Type, member roles and navigational structure defined with a DDL

3.2.1 HC-Data model

The HC-Data model is a semantic hypermedia data model. As such, it is less concerned with data structures and operations that may be applied to those data structures than it is with semantic entities and relationships that exist between these entities to which data structures may be assigned. Although the HC-Data model commits to an existence of a basic data structure, namely the hypermedia composite it still provides the possibility to model hypermedia database in the terms of types of composites (entities), members (entities) and roles (relationships between entities) of members in hypermedia composites. Even, the relationship between a member and its enclosing composite (“is-part-of” or “is-member-of”) may be seen as a relationship between the two entities. However, the HC-Data model does not allow defining other entity

types or different relationships between two main entities except for defining different roles for members within a composite. Although, somewhat restrictive this mechanism insures enough power to structure the hypermedia database on a meta-level while providing the integrity of the database as whole.

3.2.2 Data structures

The HC-Data Model operates on so-called HC-Units and HC-Types [Helic et al., 1999a; Helic et al., 1999b]. The prefix HC stands here for Hypermedia Composite.

An HC-Type is the definition of a new hypermedia composite type. It includes the definition of roles for members of this type, as well as the definition of the navigational (and in some cases of the composite visualization) structure. HC-Units are instances of different HC-Types. To relate to the object-oriented programming paradigm: an HC-Type is a class definition (defining properties and the behavior), whereas HC-Units are instances or objects of this class.

Whereas logical data models handle different data structures, semantic data model handles abstract data entities that have certain properties and relationships to other entities. For instance, following to this approach we can model a person as an entity “Person” that is related to the entities “Biography”, “Picture” and “Publications” with an “owner” relation. Speaking in terms of the HC-Data model we can identify an HC-Type called “Person”, that has three members with roles “Biography”, “Picture” and “Publication”. An instance of this HC-Type, i.e., an HC-Unit would be a hypermedia composite with the name “Nick” and the HTML documents describing the biography, a number of publications of “Nick” and showing the picture of “Nick”, assigned to the corresponding member roles respectively.

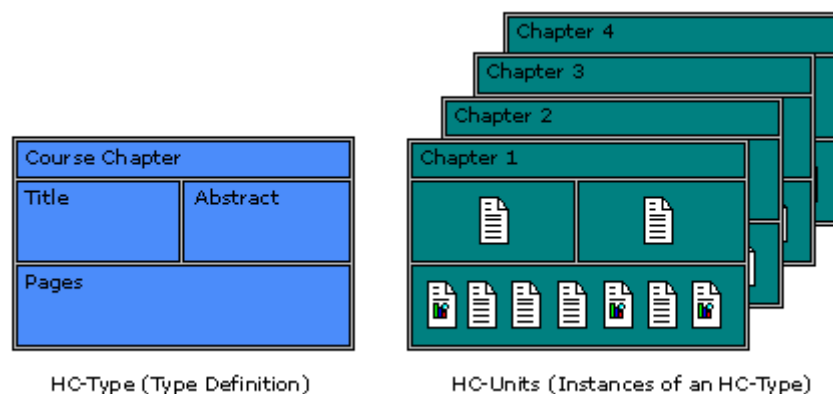


Figure 3.4 HC-Type and HC-Units

In order to define new types of hypermedia composites, i.e., new HC-Types the HC-Data model provides a Data-definition-language (DDL). The introduction of a DDL arises a need for a new user role in a particular hypermedia system, namely the role of the database administrator, who is responsible to define new types of hypermedia composites according to requirements of a particular hypermedia application or a particular author. In this way, the database administrator is able to provide application-specific or purpose-oriented hypermedia data structures, which may fulfill the requirements of any particular hypermedia application.

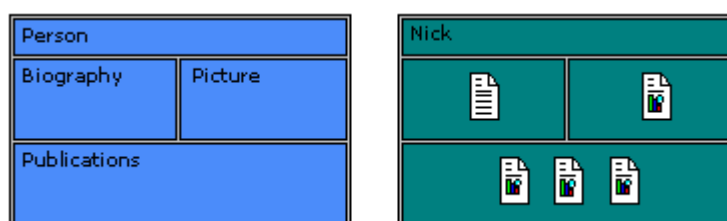


Figure 3.5 Person HC-Type and an HC-Unit

3.2.3 Defining properties of a new HC-Type

An HC-Type defines common properties that all instances of this type share. These properties are divided into:

- Properties of a hypermedia composite itself
- Properties of the members of a hypermedia composite
- So-called member roles of each particular member of a hypermedia composite.

The defined properties are attached to each instance of a particular HC-Type, i.e., to each HC-Unit of this type. The HC-Type properties appear in the form of key-value pairs. Values for some of the properties may be already specified within the HC-Type specification. Such values are automatically inherited by all HC-Units of this particular type. If the value of a property is not specified within the definition of an HC-Type such value is specified during the creation process of an HC-Unit.

There are certain properties that an HC-Type is supposed to define, i.e., their definition is obligatory. These properties include:

- `composite_type_` property with the assigned value; the value of this property specifies the entity name that this HC-Type represents (e.g. “Person”)
- `composite_name_` property without the value; the value is specified during the manipulation process of HC-Units
- `composite_navigation_` property with the assigned value; the value specifies how the composite may be navigated
- `composite_url_` property without the value; the value is specified during the manipulation process of HC-Units and specifies the Web address of each particular HC-Unit.

Also, there are member properties supposed to be defined by each HC-Type (note, that the definition of members may include the definition of one or more members):

- `member_role_` property with the assigned value; the value defines the role of a particular member within a hypermedia composite (e.g., “Publications” within the “Person” HC-Type)
- `member_name_` property without the value; same as for the `composite_name_` property – this property is specified during the manipulation process
- `member_position_` property without the value; the value is specified during the HC-Unit manipulation process and denotes the position of a particular member within the sequence of members with the same role
- `member_url_` property without the value; same as for the `composite_url_` property – this property is specified during the manipulation process and refers to the Web address of the document or another hypermedia composite that is assigned to a particular member role.

Additional properties can be defined for both hypermedia composite and/or its members with or without assigned values.

Let us now take a look on the above-mentioned example of the “Person” HC-Type. The “Person” HC-Type would have the following properties:

- `composite_type_` property with the value “Person”
- `composite_name_` property without the value
- `composite_navigation_` property with the value “person.nav”
- `composite_url_` property without the value

Note, that the `composite_navigation_` property has the “person.nav” value that is a reference to the definition of the navigational and visualization paradigm for the “Person” HC-Type. This part of the HC-Data Model Data Definition Language is explained in details in the next section.

The three members of the “Person” HC-Type may be defined with the following properties:

- “Biography” member:
 - o `member_role_` property with the value “Biography”
 - o `member_name_` property without the value
 - o `member_position_` property without the value

- member_url_ property without the value
- “Picture” member:
 - member_role_ property with the value “Picture”
 - member_name_ property without the value
 - member_position_ property without the value
 - member_url_ property without the value

Type: Person Name:	Navigation: Person, nav URL:
Role: Biography Name: Position: URL:	Role: Picture Name: Position: URL:
Role: Publications Name: Position: URL:	

Figure 3.6 Properties of Person HC-Type

- “Current Work” member:
 - member_role_ property with the value “Publications”
 - member_name_ property without the value
 - member_position_ property without the value
 - member_url_ property without the value

3.2.4 Defining navigational structure of a new HC-Type

Each HC-Type defines a particular navigational and visualization paradigm shared by all instances of this HC-Type, i.e., by all HC-Units of this type. One of the required properties included in the definition of an HC-Type is the property called `composite_navigation_`, which identifies a particular specification of the navigational and visualization paradigm.

The specification of the navigational structure and the visualization of an HC-Type consists of the specification of a number of so-called screen templates. A screen template defines what is shown on the users’ screen during each particular step of navigation through HC-Units of an HC-Type. For instance, for the above-introduced “Person” HC-Type we may specify the following screen templates:

- Screen template that is shown on the users’ screen whenever a “Person” HC-Unit is accessed; this screen template may show the documents associated with the “Picture” and the “Biography” member, the name of the HC-Unit and a list of links labeled “List of my publications” with links to each publication.
- Screen template that is shown upon the activation of a link from the “List of my publications” that shows the document associated with a particular “Publication” member of the HC-Unit and the link back to the first screen template.

Each screen template is composed of a number of so-called placeholders. A placeholder is a rectangular area within a screen template associated with a data source. Whenever a screen template is visualized on a user screen all of its placeholders are filled with the data from their corresponding data sources, and such dynamically composed document is presented on the screen.

There are several different types of data sources, such as documents that are assigned to particular member roles within a hypermedia composite, values of the properties of a particular hypermedia composite and/or its members, predefined media objects (texts, digital images, etc.) or links.

An analysis of the above “Person” HC-Type and its two screen templates (let us call them “Access” and “Current Work” screen template – see Figure 5.9) shows that the following placeholders (each of them with a particular data source) need to be defined:

- “Access” screen template:

- “Picture Place Holder” having as the data source the document associated with the “Picture” member of a particular HC-Unit
- “Biography Place Holder” having as the data source the document associated with the “Biography” member of a particular HC-Unit
- “Name Place Holder” having as the data source the name of a particular HC-Unit
- “List of my publications Place Holder” having as the data source links to the “Publication” screen template for each particular “Publication”.

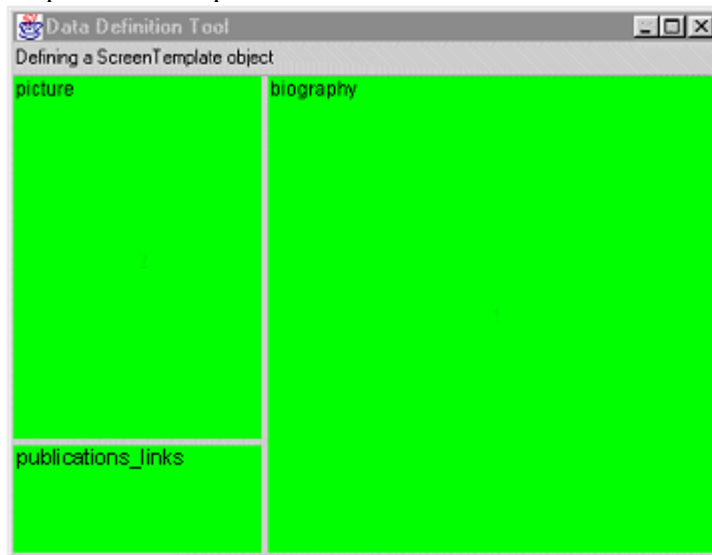


Figure 3.7 Defining Screen Template topology

- “Publication” screen template:
 - “Current Publication Place Holder” having as the data source the document associated with the current “Publication” member of a particular HC-Unit
 - “Back Place Holder” having as the data source a link to the “Access” screen template.

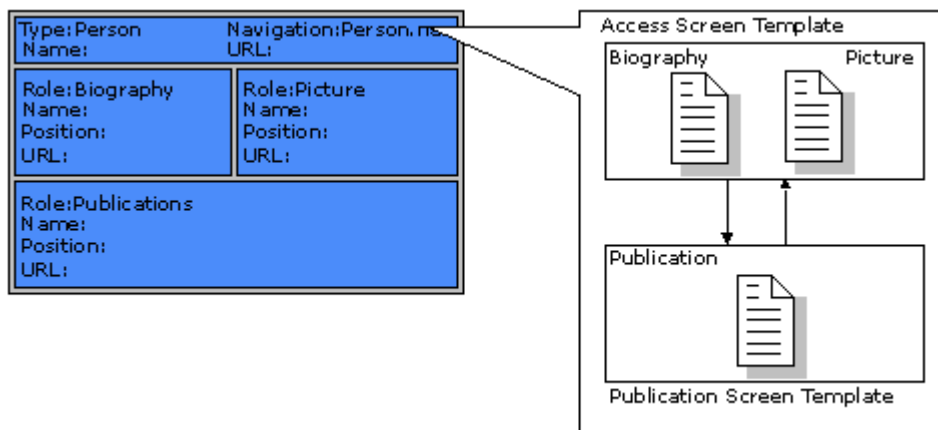


Figure 3.8 Navigational structure of Person HC-Type

The spatial arrangement of different placeholders into an HTML Page is done in the accordance to the principles of HTML authoring tool called “Page Editor” [Maglajlic et al., 1999]. Basically, “Page Editor” creates a page that is composed out of a number of rectangular placeholders, which may be arranged into a tabular representation (see Figure 3.7).

3.2.5 Browsing

The browsing in the HC-Data model is similar to that of the HM-Data model. As already mentioned, the HC-Data model inherits the basic operations of the HM-Data model: Access, Zoom-In, Zoom-Out and Zoom-Up.

Thus, whenever users visit an HC-Unit, they do not enter it, even though the HC-Unit is visualized according to its corresponding HC-Type (e.g. the “Access” screen template from the above example may be visualized). To enter it, users must explicitly use the Zoom-In operation. As it is the case with S-Collections zooming in an HC-Unit makes it become the new current container resulting in the change of the current context. Only links defined by the new current container are now accessible. In order to change the current navigational context users perform the Zoom-Out operation (which is the opposite of the Zoom-In operation), which gets them back to the HC-Unit from which the current one was accessed. The Zoom-Up operation may be also used which affects a jump from the current container to another that also contains the current one. As there may be more than one such container, the operation must also specify which.

To the contrary, the HC-Data model does not have a number of predefined navigational contexts as the HM-Data model does (e.g. Folder, Envelope, etc.) but the internal navigational and visualization paradigm is always defined within a particular HC-Type and hence inherited by all HC-Units of this type. Thus, such navigational context may differ from an HC-Type to another.

3.2.6 Creating and modifying a hyperweb

Building a hypermedia database according to the HC-Data model is similar to that of the HM-Data model. First, multimedia pages are prepared by means of stand-alone authoring tools. Those pages are building blocks for composites (HC-Units) that constitute the hypermedia database.

However, the HC-Data model does not have the predefined types of hypermedia composites, but rather those types have to be defined by means of the Data-definition-language. As already mentioned, this implies an existence of a so-called database administrator who is responsible for providing authors with the desired HC-Types. Authors according to their current needs (and requirements of a particular application) ask the database administrator to define few HC-Types providing him/her with a detailed description of a problem domain. According to such specification the database administrator is able to define the desired HC-Types.

Assuming that a number of HC-Types has been created, authors create HC-Units of those HC-Types assigning it a unique name, selecting pages or existing HC-Units as its members.

Once an HC-Unit has been created further manipulation is possible, i.e., authors may wish to insert, remove or update its members. These operations automatically update the navigational structure of an HC-Unit, as well as the way this particular HC-Unit is visualized. Note, that such operations do not alter the definition of the navigational structure of an HC-Unit, but rather because the data sources of this HC-Unit have been altered the navigational structure of the HC-Unit may be altered as well (e.g. if the document which is the data source for a particular place holder has been altered, the content of that place holder when it is visualized are automatically altered as well).

3.2.7 Mapping the HC-Data model onto the Web

Formally, a particular HC-Type is the definition of a specific navigational and visualization structure, which is automatically supported by all instances (i.e., by all HC-Units) of this type. Thus, we can say that an HC-Unit automatically imposes a particular navigable and visualization structure on the top of collections of existing HTML pages and other HC-Units defined as its members [Helic et al., 1999a; Helic et al., 1999b].

Practically speaking, an HC-Type defines a number of different member roles. Such a data structure may be simply visualized as a special template consisting of a number of rectangular cells that correspond to different member roles. Similarly, HC-Units might be seen as such template filled with existing HTML pages and other HC-Units.

From the authors’ point of view, there is a number of different templates (HC-Types) where he/she can simply insert existing pages and other HC-Units to define sophisticated navigational structures.

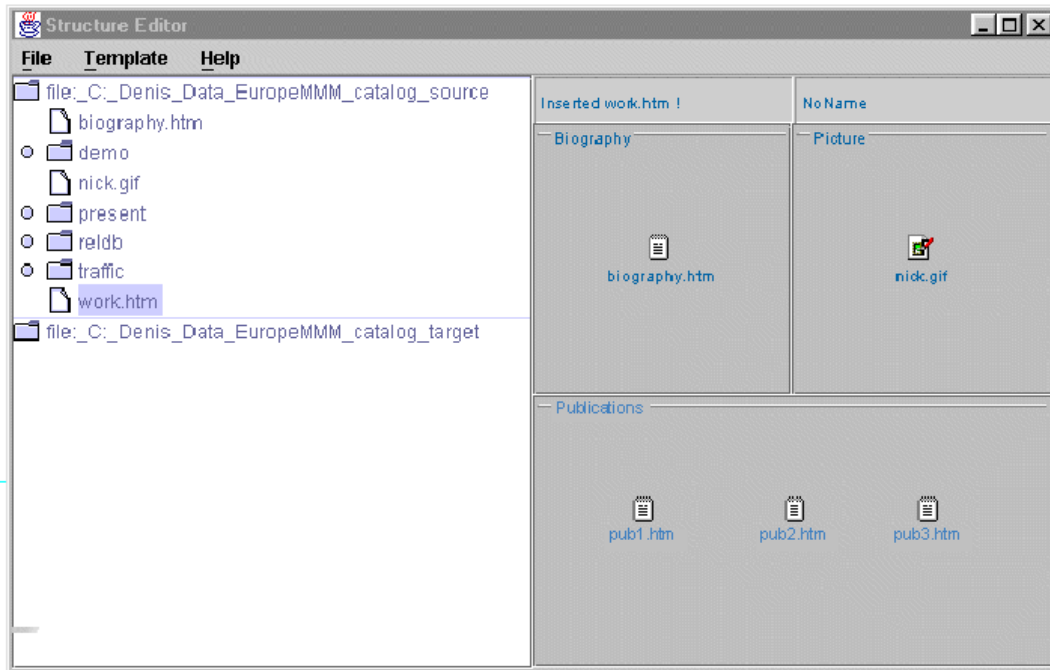


Figure 3.9 Authoring an HC-Unit

As it is the case with the HM-Data model, users access HC-Units residing on a particular Web server with their standard Web browsers such as Microsoft Internet Explorer or Netscape Navigator. However, in order to obtain the data browsers do not communicate directly with the Web server, but rather with a special rendering engine that is responsible for the mapping mechanism between the HC-Data model and the node-link data model. Note, that the newly developed Web technologies (most notably Java applets) allows that the rendering engine may be implemented even as the part of the client-side application, i.e., it may run as a Java applet within a Java-enabled Web browser.

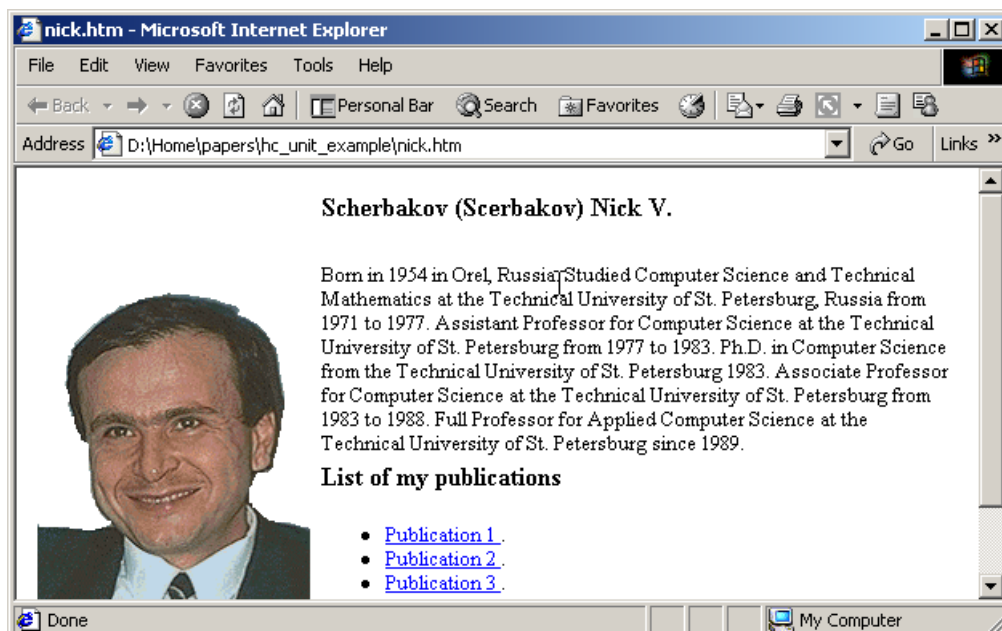


Figure 3.10 Rendering an HC-Unit of Person HC-Type

3.2.8 An application of the HC-Data model

The HC-Data model has been successfully implemented in a novel Web-based-training (WBT) [Helic et al., 2000; Dietinger and Maurer, 1997; Dietinger and Maurer, 1998] system called WBT-Master [Helic et al., 2001; Helic et al., 2001a; Helic et al., 2001b]. WBT-Master is an innovative WBT tool that supports the

construction and delivery of Internet based courseware and provides all other important WBT information services on the base of the HC-Data model. In other words, WBT-Master is an Internet system that provides a set of modules and tools that use a unified internal data structures and well-defined set of operations applicable to such data structures.

The courseware repository on WBT-Master is structured in accordance with the HC-Data model which provides for a smooth navigation through the course eliminating problems such as "getting lost in hyperspace", dangling links and similar. The model facilitates a context-dependent search and course maps. Tutors and Learners may contribute to the courseware repository "on-the-fly" using such embedded mechanisms as annotations, links to external resources and multimedia attachments. All such additional elements may be defined as public, private or visible just to a group of people, and hence provide rather powerful customization facilities.

WBT-Master supports also more traditional communicational strategies such as discussion forums, brain storming sessions, chats, exchange with private messages (ICQ). Communication may occur between learners, tutors and groups of users. Since all the communicational tools are based on the same background – the HC-Data model, any contribution may be seen as an information object, which may be stored into a courseware repository and further reused.

3.2.9 Illustrative example

Consider, for instance, a hypermedia database containing a large amount of computer-based educational material (courseware), where users can browse particular courses represented in hypermedia form. To be more specific, let us assume that we have a number of computer based courses prepared in the form of HC-Units: say "Course#1" and "Course#2" that all instances of the same HC-Type, say "Course".

Each course has a title document and a number of members corresponding to chapters of the course. If a course, let us say Course#1, refers to another course, say, Course#2, the course referred to has to be inserted into the same HC-Unit as a chapter member. A chapter consists of primitive documents and refers to other courses or chapters, i.e., a chapter is an HC-Unit of another HC-Type, say "Chapter".

It should be noted that if particular chapters and/or documents are relevant to the contents of a number of courses, the corresponding HC-Units might be re-used without any extra overhead. HC-Unit "Author-X" has a short description about the author as its title page and the author's courses (HC-Unit "Course#1", HC-Unit "Course#2", etc.) as members, i.e., the HC-Unit "Author-X" is an instance of the "Author" HC-Type.

The above discussion has indicated how primitive chunks (i.e. documents) may be gathered into more complex structures (in this case, "chapters"). The property of the HC-Data Model that an HC-Unit may belong to many other HC-Units, even recursively, provides all the necessary power to deal with more complex situations. Note the recursive membership of documents "Course#1" and "Author-X" in this example. Such recursive membership elegantly handles the common situation where a user needs to be able to access information about "Course#1" while browsing information about "Author-X" and vice versa. Moreover, if a certain chapter ("Chapter-B") refers to another course ("Course#3"), then the S-collection "Chapter-B" can be extended with the relevant member.

To conclude our example, the HC-Units representing courses might be inserted into HC-Units dealing with particular topics (say, "Topic#1", "Topic#2" etc., which are instances of "Topic" HC-Type). Finally, one could combine the entire topic HC-Units into an HC-Unit "Library of Courseware" of the HC-Type "Library".

To show the basic properties of this model, let us simulate the steps of a typical user session. Suppose the user accesses the HC-Unit "Library of Courseware" in some way. Suppose that the user zooms in the HC-Unit "Library of Courseware" and activates the link to the HC-Unit "Topic#1" within the "Library of Courseware". The HC-Unit "Topic#1" becomes the current member, and the chunk of multimedia information defined as its first screen template is displayed. Note that the navigational paradigm associated with the current document "Library of Courseware" is still active. The user has the possibility to access

another topic by clicking on it. After selecting a particular topic, the user can zoom into this HC-Unit. Once an HC-Unit is opened the user obtains (i.e., can follow) links encapsulated within it (perhaps a menu of courses).

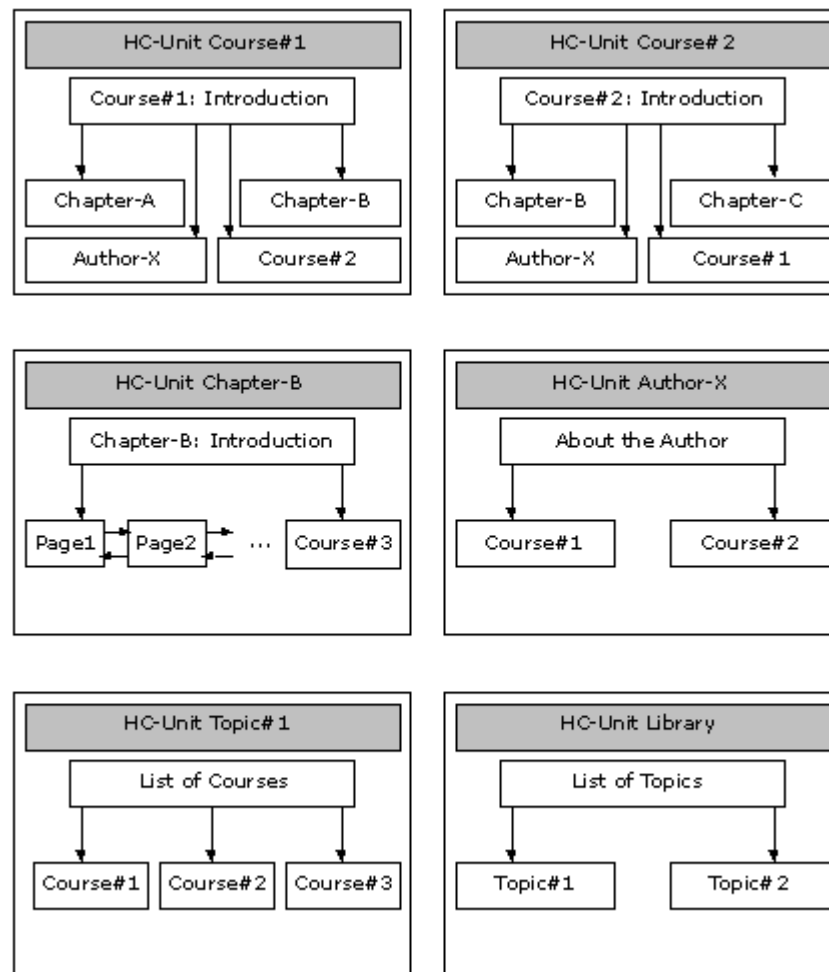


Figure 3.11 Sample hypermedia database

Each choice from the menu results in the presentation of the first screen template of the corresponding HC-Unit. If the user has selected and entered the HC-Unit "Course#1" and then selected the HC-Unit "Chapter-B", the starting screen template, say including the abstract of the chapter is visualized ("Chapter-B" is the current member) and links to other members (i.e., to other chapters) become available. The user can either read the current chapter (i.e., open the HC-Unit) or browse other chapters. Clicking on button "Zoom Out" returns the user to HC-Unit "Topic#1". Now clicking on another member of "Topic#1" (say, on "Course#2") visualizes that member's starting screen template, clicking on button "Zoom Out" leaves "Topic#1", and returns the user to the HC-Unit "Library of Courseware", and so forth.

3.2.10 HC-Types in WBT-Master

As the example above shows a number of different HC-Types is utilized by the WBT-Master. Such HC-Types are highly purpose-oriented and specific for a standard WBT application. Thus, the WBT-Master supports the HC-Types useful in dealing with data structures often seen in WBT environments such as learning units, discussion forums, brainstorming sessions and similar [Helic et al., 2001; Helic et al., 2001a; Helic et al., 2001b].

Thus, authors in order to provide learners with the educational material may use a number of HC-Types, previously defined by the database administrator. Authors manipulate the WBT-Master database, according to such particular database schema (i.e., templates or HC-Types provided by the database administrator),

thus creating instances of these HC-Types, i.e., HC-Units. These instances or HC-Units may be browsed, as defined in the navigational paradigm of a particular HC-Type, by end users of the WBT-Master, in this case by different learners.

3.2.11 Discussion of the semantic hypermedia composites approach

The HC-Data model is a generalization (an abstraction) of the HM-Data model in the sense that it extends the concept of hypermedia composites to the concept of semantic hypermedia composites providing a higher level of abstraction for different data structures involved in a particular application. Such generalization of the HM-Data model would not be possible without an extensive use of the Data-definition-language summarized through the concept of HC-Type, i.e., a definition of a class of hypermedia composites that all share one and the same properties and the behavior. Hence, the HC-Data Model does not itself define a number of different classes of hypermedia composites, but rather it gives administrators facilities to define new composite types useful for a particular application.

All that is said for the HM-Data model and its advantages over the node-link data model is still valid in the case of the HC-Data model (i.e., in the case of a generalized HM-Data model). Thus, we discuss here rather the improvements of the HC-Data model over the HM-Data model itself.

The advantages of the HC-Data Model over its logical data-modeling pendant can be stated as follows:

- Possibility to apply purpose-oriented or application-specific data structures. For example, WBT-Master, being a typical WBT application, defines a wide range of data structures typical for a WBT system. Those data structures include learning units, learning course, discussion forum and similar. Such data structures are easier to apply by all users of such a system. For instance, for authors in a typical WBT system is far more intuitive to manipulate learning units or learning courses, than it would be to manipulate S-Collections. On the other hand, learners browse the content of such hypermedia database; the possibility to browse those data structures improves the possibility of learners to comprehend the educational material in the right way to say at least. Thus, users work with data structures that are much closer to real-life objects rather than with such generic data structures as Folders, Envelopes, etc. This facilitates improved overall understanding of the purpose of a particular hypermedia system.
- Template based authoring decreases tremendously authoring effort. The concept of the definition of a type of hypermedia composite (HC-Type) provides the base for a template-based authoring. Again, WBT-Master implements such a facility on the full extent. It allows authors a rapid production of qualitative educational material.
- Inheritance mechanism allows for creating of a wide range of instances of one end the same type. All of these instances share common properties and behavior.
- Through the concept of properties that are defined in an HC-Type, documents are provided with useful meta-data. Those meta-data may be used to provide users with useful navigational tools such as meta-data search engines.

Such advantages provide means for supporting knowledge structuring process in hypermedia systems. Thus, a resultant hyperweb is composed of a number of well-structured knowledge chunks of hypermedia information

On the other hand, we might also see a number of disadvantages. Let us consider the following example. One of the basic concepts of the HC-Data model can be defined as imposing different types (HC-Types) of data structures on top of existing collections of HTML documents and/or other data structures. As it was the case with the HM-Data model, this allows a satisfactory level of reuse of documents and composites in different contexts.

However, all those previously discussed data structures such as hypermedia composites, different HC-Types, HC-Units were "navigational oriented" so to speak. They mainly can be perceived as different navigational paradigms, reflecting mainly different ways of accessing and working through a particular hyperweb by users of hypermedia systems. Primitively speaking, a "navigational oriented" data structure consisting of documents "B" and "C" mainly prescribes reading "B" before reading "C" and has nothing to do with a

possible situation that "C" may be a documentation on a software module implemented by the programmer "A" for a project "B". Often, users need a general overview and access to all documents provided by a particular hypermedia system.

Let us just discuss the following situation [Helic et al., 2001a]. Suppose a software organization maintains a big repository of technical documents in the form of a WWW application. Obviously, elements of the repository are valuable resources and may be reused as components of different composites. At the same time, localization of a particular document may constitute a rather difficult problem which can be only solved by structuring the repository on meta-level invariantly to any navigational paradigm. Using the same primitive language as before, we can say that the knowledge: "C" is a technical description of the software module implemented by the programmer "A" for the project "B" should be kept independently of reusing "C", "B" and "A" in different contexts.

As we see, a possibility to define richer network of entities and relationships between such entities, which is absent in the HC-Data model may be seen as being of primary importance. As already mentioned, HC-Units and their members may be seen as entities, which are related by means of the "is-part-of" relationship. Often, in order to model a particular hypermedia database on the semantic level we need to say more about the contents of this database (e.g., "C" is a technical description of the software module implemented by the programmer "A" for the project "B").

Actually, the described situation reflects the process of knowledge profiling (as described in the Section 2.10.2). Thus, the HC-Data model is not expressive enough to provide means for creating a hyperweb with profiled knowledge attached to it.

3.3 Knowledge Domains: Knowledge profiling in hypermedia systems

As described above (section 2.10.2), the process of knowledge profiling in hypermedia systems cannot be supported by means of logical hypermedia data modeling concepts. Further, the evaluation of advantages and disadvantages of the concept of semantic hypermedia composites has shown a number of important results: hypermedia composites, even though they provide means of knowledge structuring in hypermedia systems, are not capable of supporting the process of knowledge profiling by means of hypermedia composite. Rather a more powerful knowledge representation technique should be applied.

3.3.1 Knowledge representation in hypermedia systems

First of all, the possibility to define new types of composites, with a possible different member roles and navigation and visualization paradigms may be seen as a great advantage of semantic hypermedia composites over their logical pendant. Such possibility provides for a resulting hypermedia database structured in accordance to the application-specific or purpose-oriented data structures, i.e., in accordance to the previously defined composite types. A hypermedia database structured in such manner is much more easier created, maintained, or accessed/browsed by a wide range of different users.

Unfortunately, a hypermedia database based on the concept of semantic hypermedia composites has also some limitations. A most important limitation of such an approach is the lack of facilities to express the semantic knowledge implicitly contained in such a database. In other words semantic hypermedia composites allows to create data structures that are highly "navigation oriented". That means that such an approach is navigation centered, so to speak. The navigational structure is put into the center, either as the matter of the authoring process (i.e., the navigational structure should be easily created and maintained) or as the matter of the information retrieval process (i.e., the navigational structure should be intuitive and easily comprehended). However, semantic hypermedia composites do not provide a possibility to structure the hypermedia database at a global semantic level. It is not possible to provide users with a general overview of the hypermedia database, i.e., with a profile of knowledge contained in such database. For instance, semantic hypermedia composites does not allow to express the fact that the document "C" is a technical description of

the software module implemented by the programmer "A" for the project "B", but merely they prescribe that the document "B" should be read before the document "C".

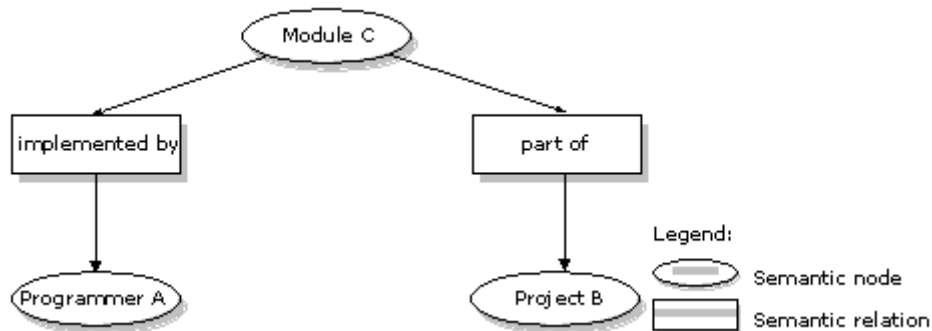


Figure 3.12 Semantic network of resources in a hypermedia database

Such a global semantic structuring may be accomplished only by means of more powerful knowledge representation mechanisms, which include a conceptual structuring of the subject matter. Such mechanisms usually support the definition of so-called domain ontology, which defines concepts, attributes and relationships between such concepts in accordance with a particular problem domain. The hypermedia database is structured corresponding to a number of such definitions, where particular document and/or composites from the hypermedia database are assigned to an arbitrary number of concepts and relationships. The outcome of such structuring technique might be for instance represented by means of network knowledge representations in the form of a semantic network of hypermedia nodes containing a semantic knowledge about hypermedia nodes and relationships between such nodes. Such graphical representations [Gains and Shaw, 1995] facilitate greatly information retrieval techniques usually applied in hypermedia systems: browsing [Gains and Shaw, 1995; Brusilowski and Schwarz, 1997; Barwise and Etchemenedy, 1990; Chang et al., 1986]. Of course, querying of such data structures [Arents and Bogaerts, 1996; Comai et al., 1998] might be also easily facilitated.

To facilitate such a global semantic structuring mechanism for hypermedia systems a concept of so-called Knowledge Domains was developed. This concept is implemented as a part of the innovative WBT system called WBT-Master.

3.3.2 Knowledge Domains

Generally, a knowledge domain is a special hypermedia-structuring paradigm, which is based on the concept of separating structure and content.

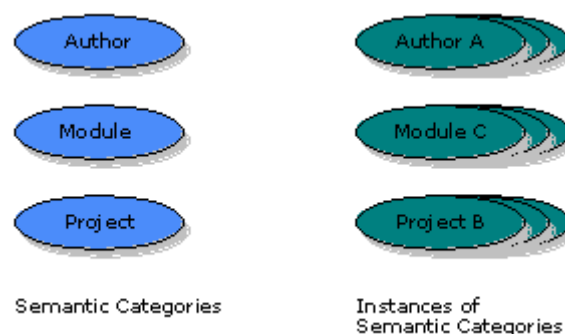


Figure 3.13 Semantic categories and their instances

Each Knowledge Domain is a set of documents and/or hypermedia composites (shortly hypermedia resources) belonging to a number of predefined semantic categories. For the previously discussed example, we could speak about three semantic categories: "Author", "Module" and "Project". We can also say that a document "C" is an instance of the category "Module", document "B" is an instance of the category "Project" and the document "A" is an instance of the category "Author". Speaking in general terms, we can

say that each semantic category is linked to a set of hypermedia resources, which are called instances of the category.

Further, each Knowledge Domain explicitly defines a number of so-called semantic relationships between the semantic categories contained in this Knowledge Domain. Again, for the previously discussed example we could speak about the following semantic relationships: "Author Modules" is a relationship between the "Author" and "Modules" semantic categories, which denotes that a particular "Author" implemented corresponding "Modules"; "Project Modules" relationship relating the "Project" and the "Module" semantic categories and denoting that a particular "Module" is a part of a certain "Project". Applying the concept of semantic relationships to instances of semantic categories means interrelating these instances according to relationships between their corresponding categories. Now, we may say that the document "C" is related to the document "A" by means of the "Author Modules" semantic relationship. Similarly, the document "C" is related to the document "B" by the means of the "Project Modules" semantic relationship.

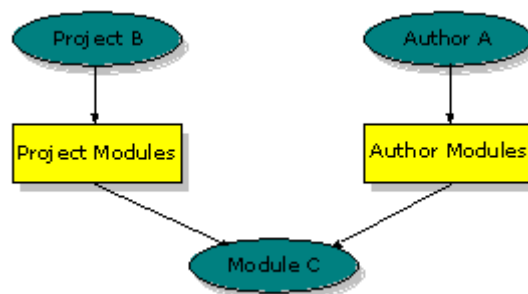


Figure 3.14 Semantic relationships

Thus, each particular knowledge domain is a collection of hypermedia resources (documents, composites, etc.), which are structured using a predefined template called the knowledge domain schema.

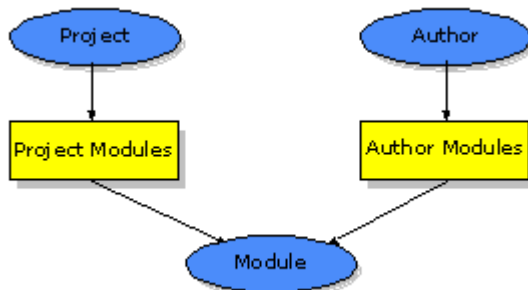


Figure 3.15 Knowledge Domain schema

A knowledge domain schema may be seen as a definition of all categories and all possible semantic relationships between them. In other words a knowledge domain schema is the definition of domain ontology.

Users may browse and search hypermedia resources, reused and interrelated with other hypermedia resources, as instances of semantic categories.

3.3.3 Data structures

The concept of Knowledge Domains operates with the following data structures:

- Knowledge domain schema
- Semantic category
- Semantic relationship
- Instances of semantic categories
- Semantic networks of category instances

A knowledge domain schema is the definition of a number of semantic categories, their attributes and relationships between different categories. In other words a knowledge domain schema is the definition of particular domain ontology. Ontology is defined as a specification of conceptualization, i.e., ontology defines concepts, properties and relationships between concepts inherent to a particular domain.

A semantic category is the definition of a particular concept from a particular domain. The name of a semantic category usually describes this concept. A semantic category has a number of attributes called data items. A data item is a standard key-value pair. Each data item has a particular type, i.e. it can be a string, a text (long string), a number and a selection from a list of values. For example, the "Author" may have just one associated attribute: Name (String). Similarly, the category "Module" may have two associated attributes - Programming Language (selection from a list of languages) and Name (String). Looking from another point of view, we can say that each instance of the category "Author" is provided with attribute "Name".

A semantic relationship defines a 1:n relationship between instances of two different semantic categories. The definition of a semantic relationship includes the selection of two categories, which participate in the relationship. A category participating with a single instance in each relationship is called "Owner" of the relationship. A category participating with multiple instances in each relationship is called "Member" of the relationship. Each instance of a Member is obligatory related to a certain instance of Owner; this reference is installed whenever a new Member is created. For instance, in the example above, the relationship "Project Modules" may relate a single instance of "Project" to an arbitrary number of "Modules", or looking from another perspective, each instance of "Module" is obligatory related to a certain "Project". This relationship will be created whenever a new "Module" instance is put into the Knowledge Domain. Similarly, the relationship "Author Modules" relates a single instance of "Author" with an arbitrary number of instances of "Module".

Each knowledge domain schema provides a number of attributes (data items) which should be provided for any instance of a particular category and types of relationships which should be installed between the new instance and existing instances of other categories.

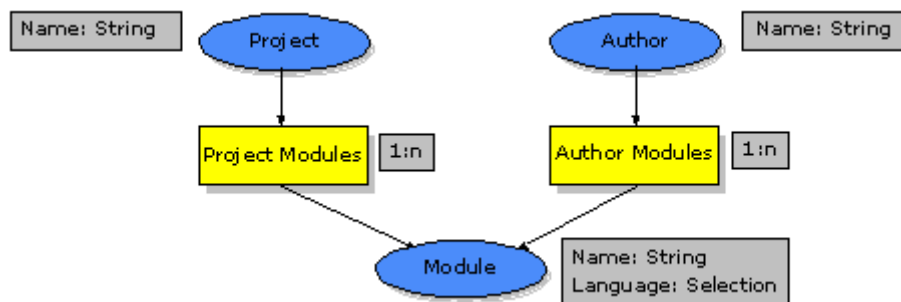


Figure 3.16 Categories, data items and relationships

The resulting data structure (the resulting hypermedia database) has the form of a semantic network, i.e., a directed labeled graph where the hypermedia resources, assigned to the nodes of the graph are interrelated by means of the semantic relationships with other hypermedia resources. Such data structure is general enough to provide a global overview of resources contained in a hypermedia database.

3.3.4 Defining a Knowledge Domain Schema

As already stated, a knowledge domain schema defines a data structure in the form of so-called semantic categories and semantic relationships between these categories. Any training resource added to the knowledge domain is perceived as an instance of one particular category and, thus, inherits all properties defined for the category.

The definition of a knowledge domain schema includes the following steps:

- Definition of a number of semantic categories (for each semantic category a number of data items is provided)
- Definition of semantic relationships (for each semantic relationship two semantic categories are provided, i.e., the owner of the relationship and the member of the relationship).

Existing of knowledge domain schema facilitates three rather important processes:

- Structuring of a knowledge domain
- Browsing a knowledge domain.
- Searching in a knowledge domain

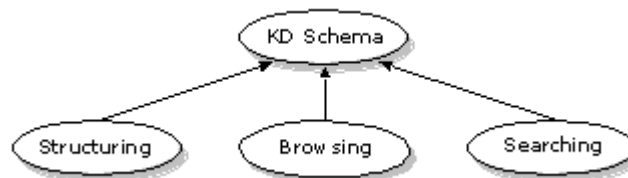


Figure 3.17 Data manipulation processes supported by a schema

Whenever a new instance of a category is created, the system may assist the user with offering special forms for providing attribute values and for selecting other instances to be related to the new one. The system may also check whether all necessary information is provided and prompt the user if necessary.

From the browsing point of view, the existence of a knowledge domain schema provides very useful information on a current document position within the knowledge domain and on semantic of links emanating from or pointing to the document [Gains and Shaw, 1995]. Further, many visualization techniques might be applied to visualize such data structures [Gains and Shaw, 1995; Brusilowski and Schwarz; 1997].

Searching in a knowledge domain may be carried out at a meta-level [Arents and Bogaerts, 1996; Comai et al., 1998; Decker et al., 1999]. The data items of different semantic categories may be seen as the meta-data, i.e., data about data. The existence of a knowledge domain schema provides for structured querying possibilities [Decker et al., 1999]. For instance, we may search for all “Modules” written in the Java Programming Language and developed by the “Author” X.

3.3.5 Creating and modifying a database

Usually, building a hypermedia database follows a bottom-up approach. First a number of hypermedia resources is prepared (i.e., hypermedia documents, hypermedia composites, semantic hypermedia composites, etc.). Then a number of knowledge domain schemas is defined, defining common properties and relationships for all instances of different semantic categories. Finally, hypermedia resources are assigned to different semantic categories, becoming instances of these categories and inheriting their properties and relationships with other instances.

Thus, a responsible author simply selects an existing knowledge domain and a predefined category for a new resource and the system guides the author through the process of defining attributes and necessary relationships. For example, if a new instance of the category "Module" is created, the system automatically request to select a programming language (attribute predefined for the category), and to provide references to the module author and a certain project (relationships predefined for the category). This, of course, facilitates creating of well-structured repositories.

As already mentioned the resulting database is a semantic network of hypermedia resources.

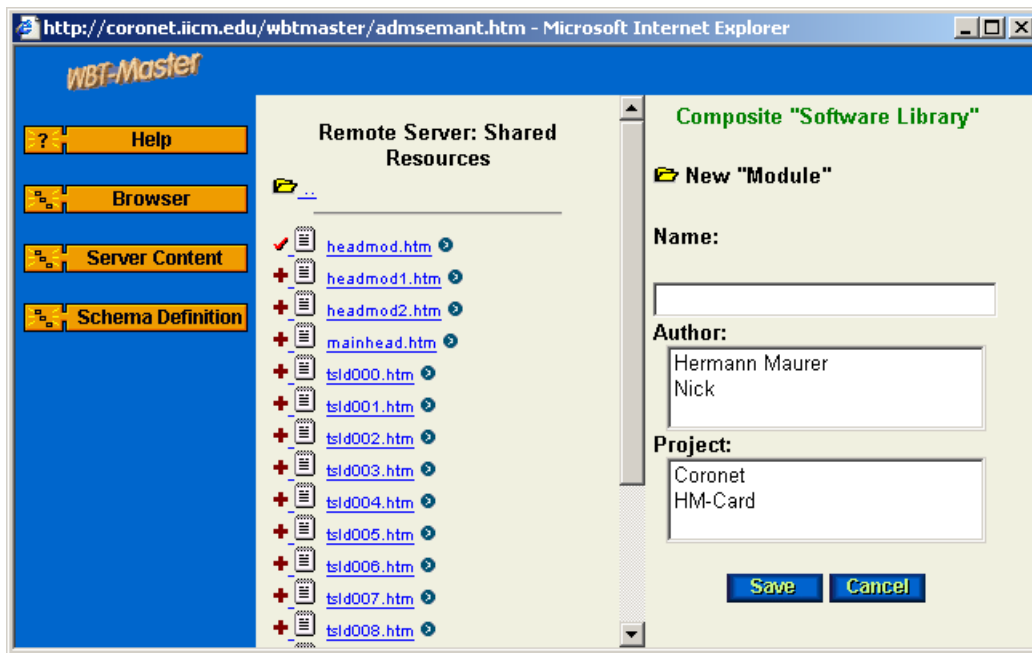


Figure 3.18 Creating an instance of a semantic category

3.3.6 Browsing

The concept of well-structured Knowledge Domains facilitates browsing carried out on a meta-level (meta-browsing). Users browse the concepts and their relationships as opposed to browsing of individual documents (hypermedia resources) [Brusilowski and Schwarz, 1997].

Information contained in a knowledge domain (i.e., categories, data items, relationships) may be applied in a number of useful ways. System may provide useful navigational tools, such as prior/next buttons, preview buttons for the related instances and similar.

Thus, for example, whenever a user access the document "Module 01/01", the system automatically provides:

- Information on attributes attached to this document
- References to instances of other semantic categories which are related to this one
- Next/prior navigational tools, etc.



Figure 3.19 Browsing an instance of a semantic category

Browsing a knowledge domain does not require any additional knowledge besides understanding of the predefined structure of the domain. Users simply select a particular knowledge domain and a certain category and all existing instances are displayed by the system. Selecting a particular instance results in the start of the actual browsing of the knowledge domain.

3.3.7 Searching

Knowledge domains facilitate powerful searching capabilities. The existence of a knowledge domain schema allows for searching in a well-defined context, by using predefined categories, data items and relationships.

For instance, the system may provide a possibility to select a semantic category, to define values for data items, to select a semantic relationship, etc.

Searching results may be seen as a very powerful tool to facilitate a so-called “initial access” to resources, i.e., to access instantly a desired hypermedia resource without browsing the whole hypermedia database.

Similar to browsing of knowledge domains, searching of knowledge domains does not require any additional knowledge besides understanding of the predefined structure of the domain. Users simply select a particular knowledge domain; provide a number of search criteria and all corresponding instances are displayed by the system.

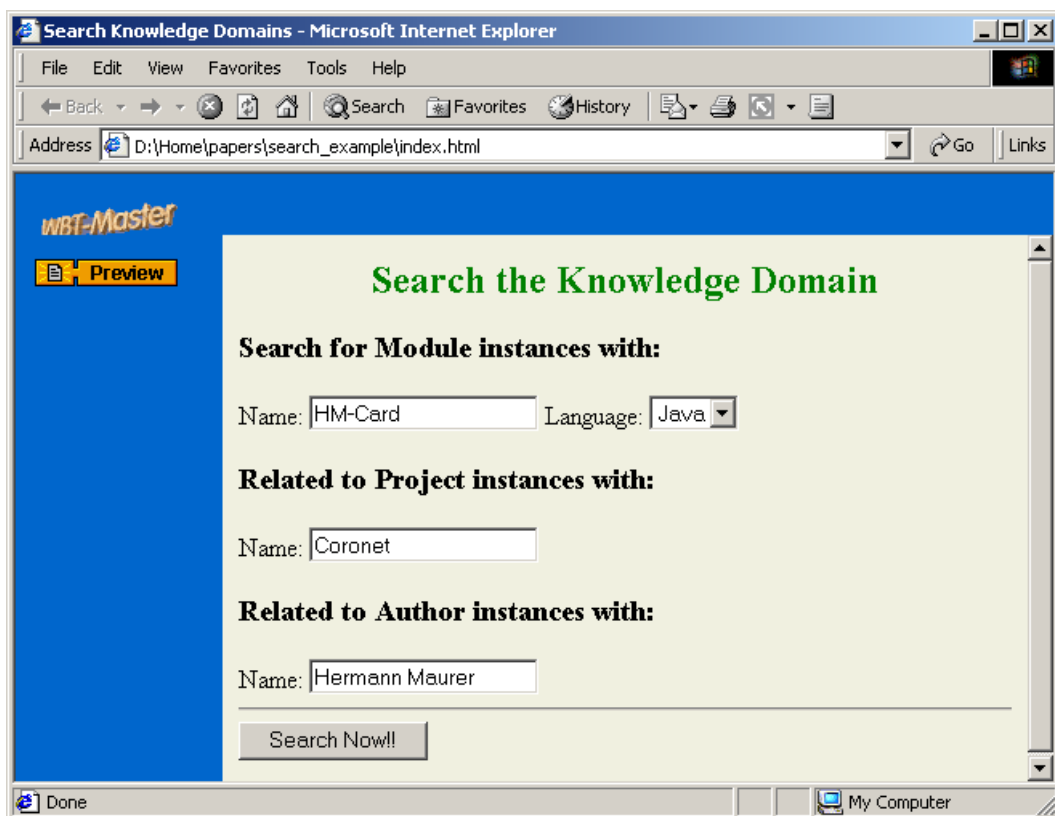


Figure 3.20 Searching in a Knowledge Domain

3.3.8 Discussion of the Knowledge Domains concept

The concept of the Knowledge Domains facilitates many important features such as:

- Separating of the structure and the content of the resulting hypermedia database - as shown many times before [Helic et al., 1999] [Helic et al., 1999a] [Maurer et al., 1996], the crucial point for the successful maintaining of large WWW based hypermedia databases. This feature might be seen as an advantage over standard knowledge representation techniques. Such feature utilizes the

prescriptive aspect of semantic data modeling approach, rather than just the descriptive aspect of traditional knowledge representation.

- Defining of new data structuring facilities through the concept of the Knowledge Domain Schema - provides a possibility to reuse hypermedia resources in different well-defined contexts, thus, allowing learners to have new views on the learning material collected in a Web system.
- The concept of the Knowledge Domain Schema provides means for template-based authoring - through the template-based authoring the authoring process is considerably simplified. A wizard-like authoring system is imaginable, as it is already implemented in the WBT-Master system. Such authoring system is easy to use, and a rapid production of hypermedia material is easy to achieve.
- The result of the authoring process is a well-structured repository of hypermedia resources - the concept of the Knowledge Domain Schema defines a database schema that is applied by authors in the authoring process to structure hypermedia resources. Thus, the result of this authoring process is a very well structured database created in the accordance to a number of predefined database schemes. Moreover, the resulting database is equipped with a global overview (i.e., with a knowledge profile) of hypermedia resources contained in the hypermedia database. Such global overview comes in the form of a semantic network.
- Searching in the system on a meta-level - attributes attached to hypermedia resources, as defined by the Knowledge Domain Schema, may be comprised as the data on about the data, i.e., the meta-data. Such meta-data may be used to enhance searching facilities provided by the system, especially when they are predefined, as it is the case with the Knowledge Domains.
- Browsing hypermedia resources on a meta-level - browsing of Knowledge Domains facilitates browsing of the concepts (comprised in semantic categories) and relationships between these concepts (comprised in semantic relationships between different categories) as opposed to browsing of individual documents or hypermedia resources. Furthermore, when browsing instances of semantic categories those instances are provided with very useful additional information contained in attribute values attached to those instances.
- Inheritance as supported by Knowledge Domain concepts might be seen as a great advantage of such an approach. It allows for modeling of a hyperweb according to a predefined database schema accompanying hypermedia resources of the same kind with one and the same properties and behavior. Again, this facilitates creating of a well-structured hyperweb.

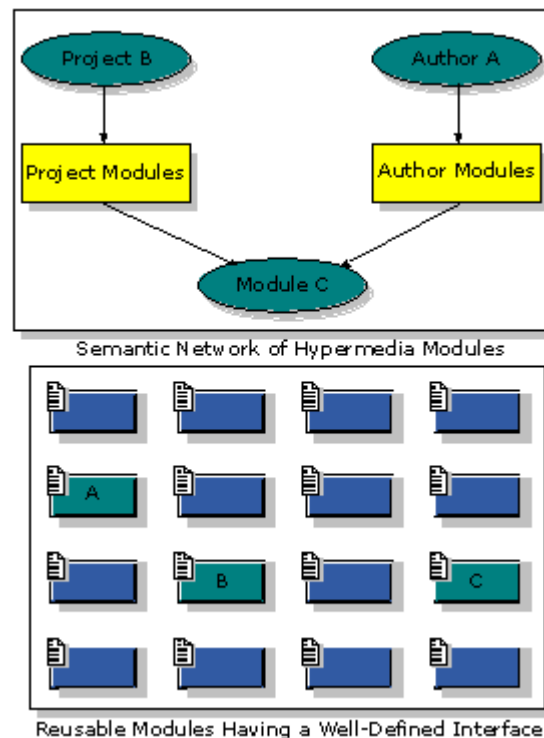


Figure 3.21 Resulting hypermedia database

However, we believe that the concept of Knowledge Domains, as it is implemented in the WBT-Master, requires a number of improvements. Those improvements are mainly concerned with the concept of semantic relationships:

- It should be possible to define not only relationships of 1:n character but also relationships of m:n character; often the 1:n relationships are not sufficient to describe real-life relationships between two concepts. For instance, an instance of the category "Module" (i.e., a software module) may be a part of not just one "Project", but possible two or more "Projects". This facilitates the description of the concept of reuse of software modules, which is one of the basic concepts in the software engineering field
- It should be possible to define not only binary relationships, but also relationships of an arbitrary arity. Usually, to describe relationships between different concepts, we need to apply n-ary relations between those concepts. For instance, if we say that the category "Module" has an attribute "Name" of the type String, we may comprehend this fact as a relationship between concepts "Module" and "Name". If we now consider other relationships of the "Module" category (e.g., "Project Modules") we end up with the category "Module" that constitutes a 3-ary relationships between categories "Name", "Module" and "Project". However, this facility should be explicitly supported by the system in order to improve the expressive power of the Knowledge Domain concept.

3.4 Knowledge Cards: Knowledge mining in hypermedia systems

The above-described process (Section 2.10.3) of knowledge mining in a hypermedia database may be easily simplified by an application of semantic network principles for providing a conceptual map of hypermedia resources and providing these semantic networks by a set of simple inference rules [Sowa, 1984; Sowa, 1991]. Such inference rules might be used to automatically infer hypermedia resources for a particular concept, thus allowing for an initial access to best-match hypermedia resources. Such approach has been successfully implemented through the concept of so-called Knowledge Cards.

3.4.1 Properties of semantic networks

A hypermedia database structured accordingly to a knowledge domain schema is a repository of well-structured reusable hypermedia modules enhanced with a semantic network of those modules. The semantic network represents knowledge contained in those hypermedia modules in the form of different concepts, to which hypermedia modules may be assigned, and relationships between these concepts.

Generally, semantic networks may be used as a component of two different information-processing tasks:

- Information Retrieval:
 - o User Interface: Semantic network may be seen as a simple and intuitive visual form of knowledge representation and, thus, as a metaphor for man-machine communication [Sowa, 1984; Sowa, 1991].
 - o Querying: Semantic network may be seen as a database schema that allows for querying the concepts and their relationships [Sowa, 1984; Sowa, 1991].
- Inference Engine: Semantic network may be seen as a special formalism for defining rules of inference [Sowa, 1984; Sowa, 1991].

A Knowledge Domain apply a semantic network to create a convenient and intuitive graphical user interface for browsing the hypermedia database, as well as a powerful search mechanism to search for hypermedia modules represented by concepts from a particular semantic network.

However, Knowledge Domains does not make use of the other very important property of semantic networks, i.e., the possibility to infer a new knowledge from the semantic network by an application of so-called inference rules.

The concept of knowledge cards tries to make use of the inference power of semantic networks. Furthermore, this concept tries to combine the information retrieval facilities of semantic networks with an inference engine based on a number of very simple inference rules.

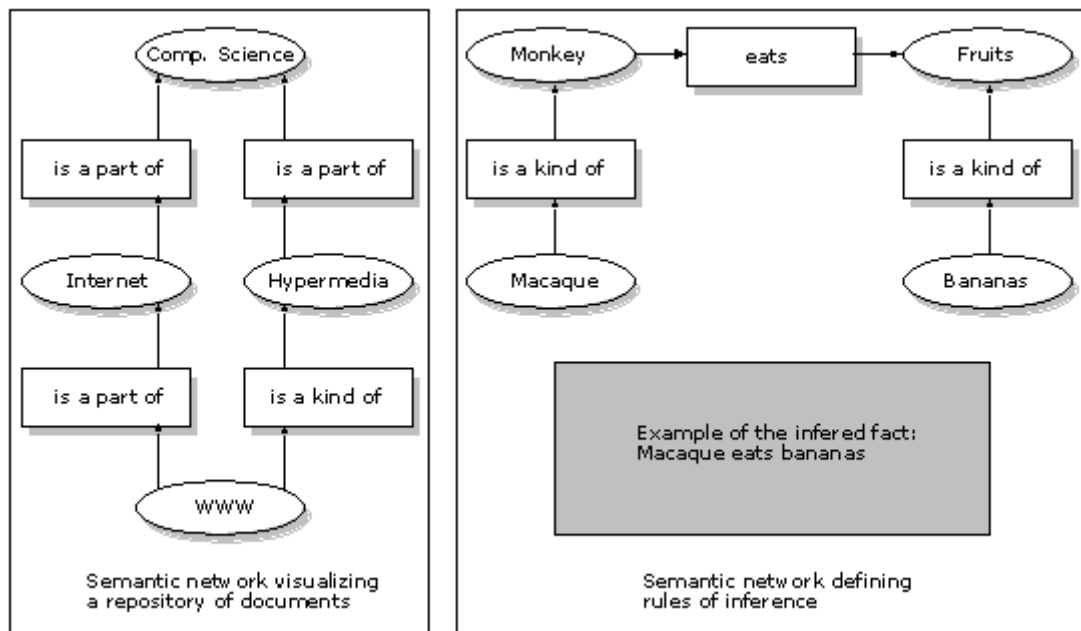


Figure 3.22 Different applications of semantic networks

The knowledge card concept has been successfully implemented in the WBT-Master.

3.4.2 Knowledge Cards

A *Knowledge card* is a description of particular concept (i.e. *semantic entity*). For example, a semantic entity “Database technology” may be seen as a knowledge card.

Practically speaking, each Knowledge Card may provide access to a number of associated hypermedia modules from the underlying hypermedia database. For example, a hypermedia module “Relational Data Model” may be associated with the Knowledge Card “Relational Data Model”, some other hypermedia documents, composites, etc. may be associated with the same Knowledge Card.

Knowledge cards are interrelated into a semantic network using different types of relationships: “is a part of”, “is a kind of”, “synonym for”, etc. For example, the knowledge card “Relational Data Model” may be related as “is a part of” to the knowledge card “Database Technology”. The knowledge card “World Wide Web” may be related as “is a kind of” to the knowledge card “Hypermedia Systems”. The knowledge card “Web Base Training” may be related as “is a synonym for” to the knowledge card “Computer Supported Collaborative Learning”.

All possible semantic relationships are predefined in a special system thesaurus where the relationships are also provided with a numeric assessment of so-called inference probability. To explain the meaning of the term “inference probability”, suppose two concepts (say, “A” and “B”) are interrelated with a relationship “A is a part of B”. Suppose also that there are two different hypermedia resources “rA” and “rB” associated with the knowledge cards “A” and “B” respectively. The inference probability gives a very rough assessment of two probabilities:

- Probability of the resource “rA” to be relevant for the card “B”;
- Probability of the resource “rB” to be relevant for the card “A”;

For example, for the previously discussed example of the “is a part of” relationship, the inference probability can be set as 1.0/0.5. It means, that if a course is associated with the knowledge card “Relational Data Model”, it is definitely relevant to “Database technology”. If the course is associated with the knowledge card “Database technology”, it is relevant to “Database technology” with a 50% probability. Obviously, the inference probability can be set as 1.0/1.0 for the “is a synonym for” relationship, etc.

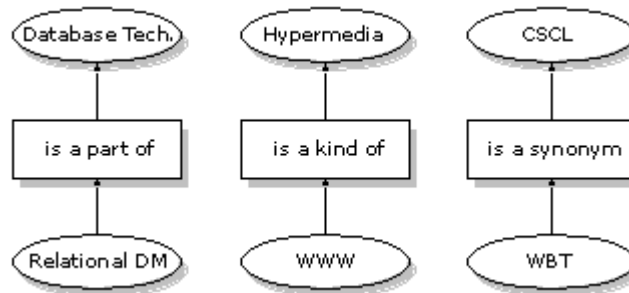


Figure 3.23 Interrelating knowledge cards into a semantic network

The semantic relationships essentially compose a graph structure (as opposed to just a hierarchical one). For example, the same knowledge card “Relational Data Model” may be defined as a part of “Introduction to Oracle”, “Information Systems”, etc. Moreover, there may be Knowledge Cards defining areas of personal interest: say “Personal Knowledge of H. Maurer” which may also refer to the previously mentioned card “Relational Data Model”, etc.

3.4.3 Creating and modifying a hyperweb

Whenever authors contribute to the system with new material, they are supposed to associate it with one or more Knowledge Cards or create a new Knowledge card and place it into a proper position within the semantic network. Of course, a specially designated member of the system administration team (Knowledge engineer) might do this as well.

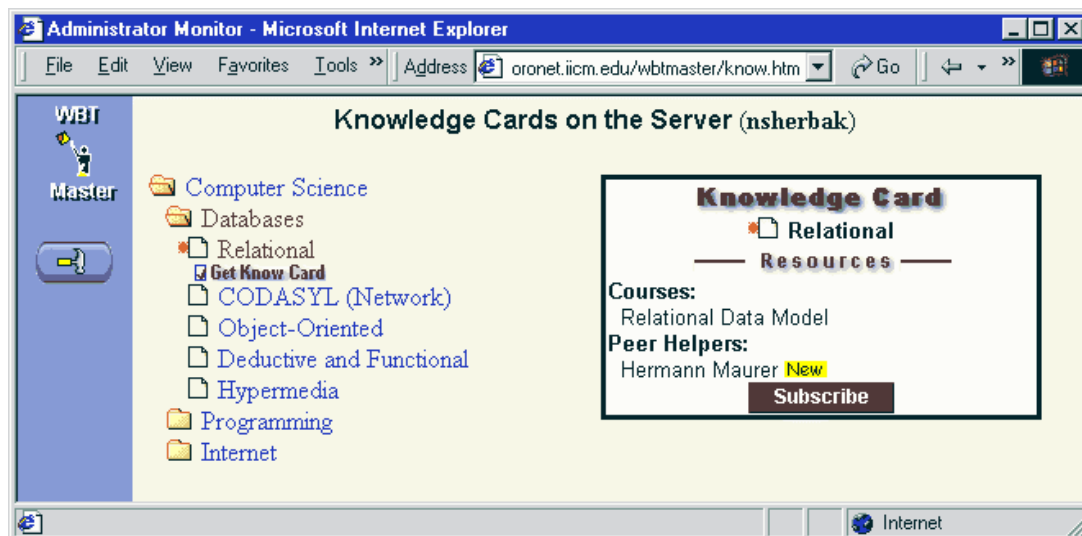


Figure 3.24 Resources attached to a Knowledge Card

3.4.4 Browsing

As a starting point, learners are not supposed to browse through countless hypermedia resources. They are supposed, in a simplest case, to browse the semantic net consisting of previously defined Knowledge Cards.

Thus, whenever a particular user accesses a Knowledge Card all hypermedia resources associated with that Knowledge Card are accessible.

Further, the system uses the inference engine in order to infer all other hypermedia resources attached to Knowledge Cards that are related to the accessed one. All such inferred hypermedia resources become accessible from that particular Knowledge Card.

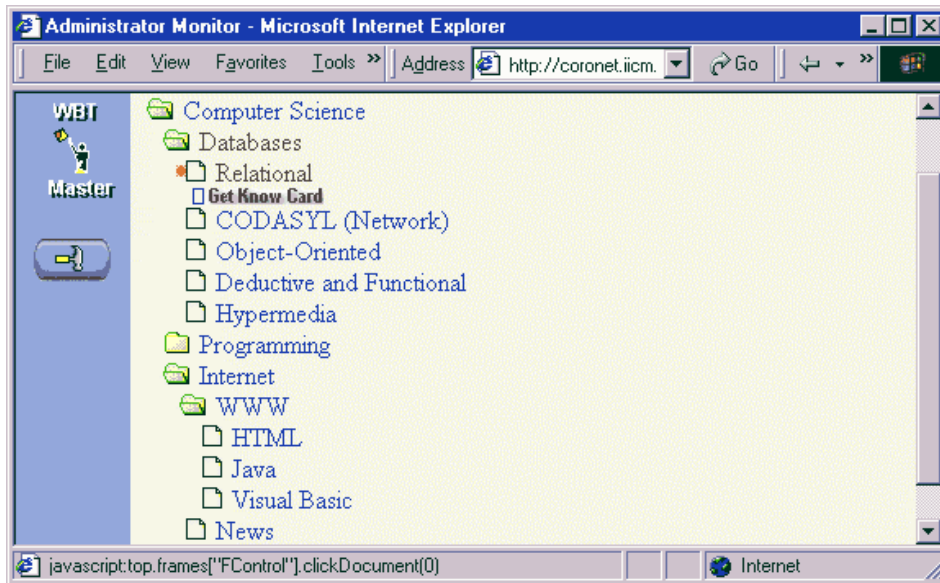


Figure 3.25 Browsing Knowledge Cards

3.4.5 Inference engine

As it was mentioned early, the mechanism essentially utilizes the other important property of the semantic network - a possibility to infer hypermedia resources using semantic relationships [Sowa, 1984; Sowa, 1991; Helic et al., 2001].

Whenever a user access a knowledge card, the system automatically infers all hypermedia resources which are associated with this particular Knowledge Card and with Knowledge Cards related to this one.

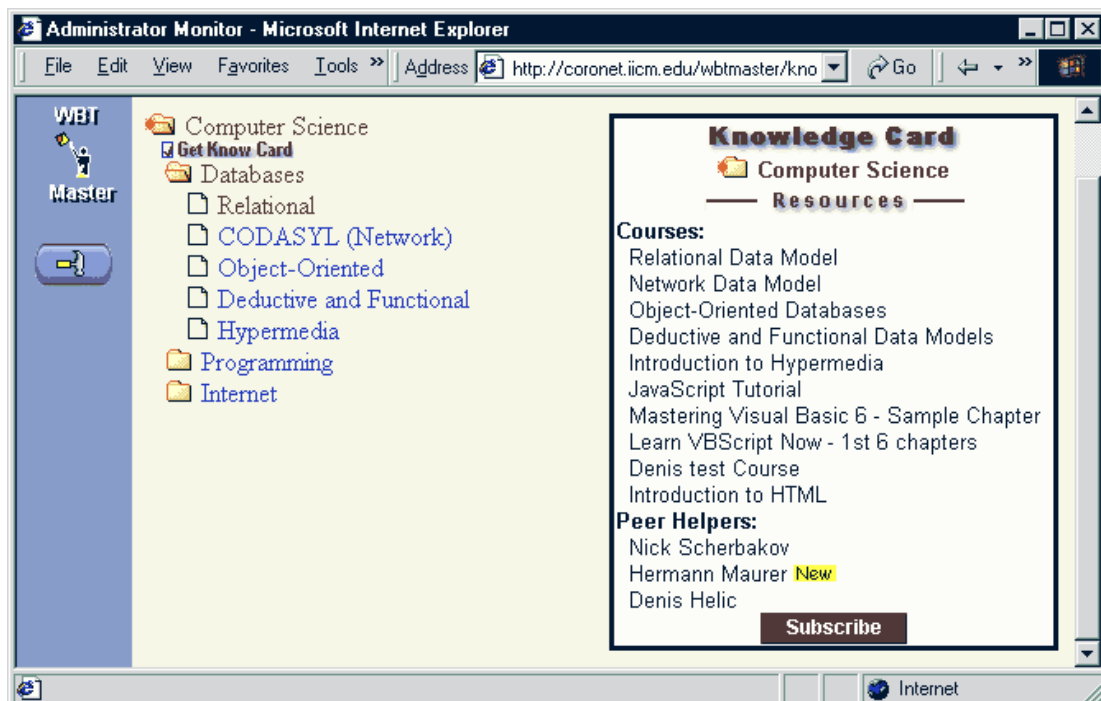


Figure 3.26 Resources automatically inferred for a Knowledge Card

The infer procedure is rather straightforward, and can be defined as follows:

- Step1: Set list of relevant knowledge cards as the current card, assign to the current card a special "relevance" attribute as 1.0.
- Step2: Get list of resources from the list of relevant knowledge cards and provide them with corresponding "relevance" attribute.

- Step3: If a representative number of resources is not inferred, replace each element of the list of relevant knowledge cards, with a set of cards related to this one. Calculate the “relevance” for each new card as:

$$\langle \text{“relevance” the card to be replaced} \rangle \times \langle \text{infer probability set for the relationship} \rangle$$
- Repeat Step2.

For example, suppose that there were no resources associated with the knowledge card “Computer Science”, but a number of other card (say, Databases, Programming, etc.) were defined as “is a part of” Computer Science. Accessing the “Computer Science” knowledge card will result in the resources inferred from other related cards with the “relevance” attribute equal to 1.0.

3.4.6 Discussion of the Knowledge Card approach

The situation discussed above, leads us to a number of very important conclusions:

- Content providers do not need to search for a precise “knowledge card” to associate their resources with. They can simply define their own field of interest (say, “Personal Knowledge of H. Maurer” and associate all their resources with this card automatically). Other users may decide that “Personal Knowledge of H. Maurer” is an important contribution to the “Hypermedia” concept, and create a relationship between these two knowledge cards. Of course, “Personal Knowledge of H. Maurer” may be further structured as a number of knowledge cards (say, “Maurer & Theory”, “Maurer & Hypermedia”, etc.), which are related “as a part of” to “Personal Knowledge of H. Maurer”.
- Learners do not need to browse the whole semantic net, they might be interested to define a personal knowledge card related to a number of previously defined concepts which are of interest. In this case, whenever users access such personal knowledge card, all relevant resources will be inferred automatically. This greatly facilitates the above-described process of knowledge mining.
- Moreover, users may adjust the semantic net by defining new types of relationship having different “infer probability” values. For example, there may well be relationships “I am very interested in”, “My main field of contributions”, etc.

Personal knowledge card may serve as a “customized” entry point to the server resources; the server may also automatically notify the learner about new resources, which might be of interest.

Generally, first experiments with the Knowledge Card system demonstrate a rather good functionality and acceptance by users. User evaluation questionnaire and analysis of the server Log files show that users definitely prefer a personal knowledge card to be an entry point to the whole system. Authors generally like the situation when they can simply contribute to the server without paying much attention to accessibility of the material. Note, the contribution is automatically associated with the author’s personal knowledge card that, in turn, refers to the author’s area of interest.

The only serious drawback mentioned by a number of users, is an absence of a special filtering mechanism.

Thus, users very much prefer not to simply infer all hypermedia resources relevant to a certain knowledge card, but be able to define what particular resources are of interest. For example, they would need a mechanism to define something like “get me all peer helpers who are currently online” or “get relevant chat sessions which are currently active, etc.

4 Introducing Semantic Data Structures to the WWW

The semantic hypermedia data modeling concepts that were introduced in the last chapter were successfully implemented in the novel Web-based-training system called WBT-Master [Helic et al., 2001; Helic et al., 2001a; Helic et al., 2001b]. This chapter gives an overview of implementation issues of those data modeling approaches.

4.1 WBT-Master

The WBT-Master is a novel Web-based-training system implemented on the following concept: a modern WBT system should provide a set of tools which support different knowledge transfer processes, thus allowing for a smooth transfer of knowledge from people who possess such knowledge (e.g. experts, teachers) to people who want to acquire this knowledge (e.g. learners, students) [Helic et al., 2001b]. Thus, WBT-Master provide tools that support the following knowledge transfer processes in a Web based environment [Helic et al., 2001b]:

- Web based knowledge structuring
- Web based knowledge mining
- Web based knowledge profiling
- Web based tutoring
- Web based mentoring
- Web based learning.

Thus, WBT-Master was an implementation of the previously discussed knowledge transfer processes in a Web environment. Here we provide the overview of such implementation.

4.1.1 WBT-Master architecture

Being a fully WWW compatible, WBT-Master considerably extends the standard WWW client-server architecture.

WBT-Master servers store complex, composite data objects additionally to primitive HTML documents. The data objects are typed, i.e. a data object always belongs to a particular data class, which defines all user's actions applicable to such data objects. Documents, portals, learning units, learning courses, learning goals, etc. are data objects residing on WBT-Master server.

The data objects are persistent, i.e. a WBT-Master server can apply so-called actions, which alter a current state of a data object, and results of such actions are available to other users. For example, a new member may be inserted into a learning unit, a new contribution added to forum, etc.

The data objects are reactive, i.e. an object replies to an *action* with a collection of data which depends on the current state of the data object, and user's context. For example, "get content" action addressed to a discussion forum, returns a structured collection of contributions made to this forum.

Actions are sent from an Internet client to the WBT-Master using ordinary HTTP protocol. Note that the HTTP requests contain not only the URL of a particular data object but also an action, which is to be applied to this data object. The WBT-Master server carries out the request, which results in:

- Possible modification of the target object current state
- Generating a response to the action.

The server response is visualized on the WBT-Master client side as resultant information and all further actions, which can be applied to such data object (as opposed to a visualization of passive HTML documents).

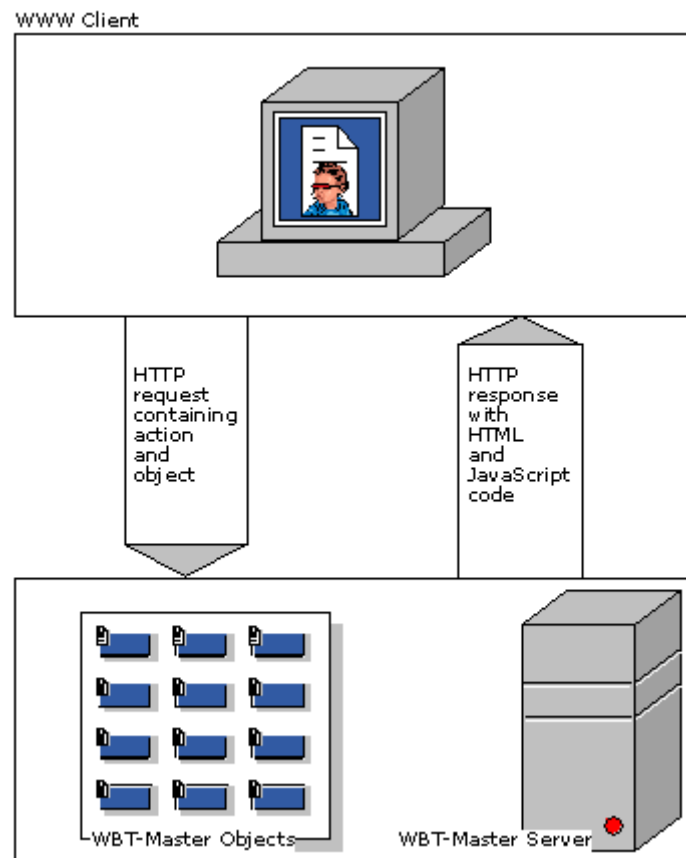


Figure 4.1 WBT-Master client-server architecture

All the previously discussed data structures, such as Knowledge Domain, Knowledge Cards or HC-Types and HC-Units are different kinds of WBT-Master objects. Thus, WBT-Master provides an implementation of the previously discussed semantic data modeling approaches.

4.1.2 WBT-Master technical solutions

Recollect the basic WBT-Master architectural principles [WBT-Master, 2001]:

- Data Objects are persistent, reside on a server and are eligible for actions that may alter the data objects.
- A particular data object belongs to one of predefined data types that define properties of the data object and valid actions.
- A client initiates an HTTP request containing reference to a target data object, an action that need to be applied to it and a number of parameters relevant to this action.
- A server applies the action to a prescribed data object. The action may affect the object's current state. The response to the action is sent back to the client where it is visualized on the user's screen.

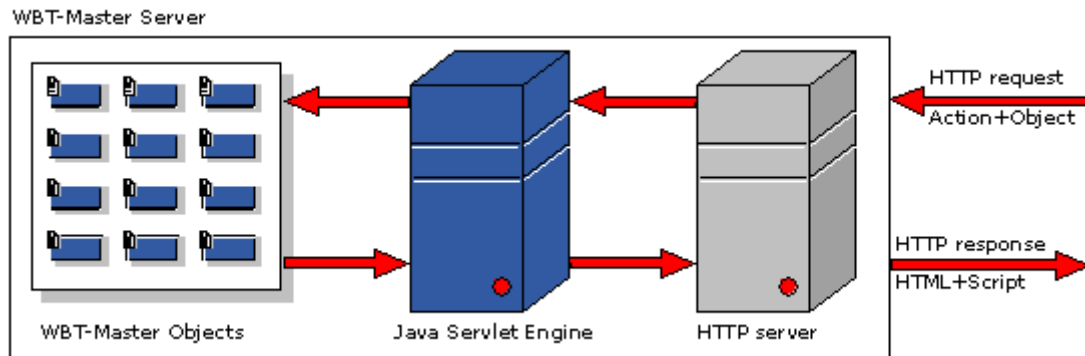


Figure 4.2 WBT-Master server architecture

Obviously, standard WWW servers and clients do not support the above-mentioned functionality.

Hence, the functionality of both components (i.e. server and client) must be extended by means of one of modern WWW programming technologies.

On the server side WBT-Master extends a standard WWW server functionality by applying so-called Java Servlet (a Java-enabled Web server). Since Java Servlet are small Java programs running in a context of a Java Servlet Engine, they provide an ideal mechanism for implementing WBT-Master actions.

Current implementation of the WBT-Master is based on Apache Web Server with the Apache Servlet module enabled and JServ Servlet Engine. However, any server that supports server site Java Applets (i.e. Servlets) can be used as well.

An Apache Web Server acts as a repository of data objects. It stores data objects in its file system. A client request including an encapsulated action is interpreted by a Servlet which actually generates a response depending on the current object state.

The current Servlets implementation assumes that data objects are instances of an abstract Java class. This basic abstract class is defined through its method interface that represents actions that can be applied to data objects. Subclasses of the basic abstract data object class support particular logic of an action applied to a predefined type of a data object.

This level of abstraction enables dynamical mapping of logical data structures onto a variety of possible physical storage spaces. For example, an ordinary Apache file system may be replaced with a database by implementing a new subclass of the basic abstract class.

Moreover, since the servlets utilize the Java Factory concept it is possible to change the physical data objects format even at the runtime.

Conceptually, an end-user always communicates with a particular instance of data object. Thus, for example, the user may work on a particular Learning course, Learning goal, Forum, etc.

On the client side, JavaScript functions and signed Java applets are used to:

- Visualize all actions applicable to a current data object;
- Convert a particularly selected action into an HTTP request;
- Visualize the action's results (i.e. server response) on the user's screen.

4.1.3 WBT-Master GUI

The Graphical User Interface (GUI) supported on the client side, is unified using a number of so-called *Functional Panels* [WBT-Master, 2001].

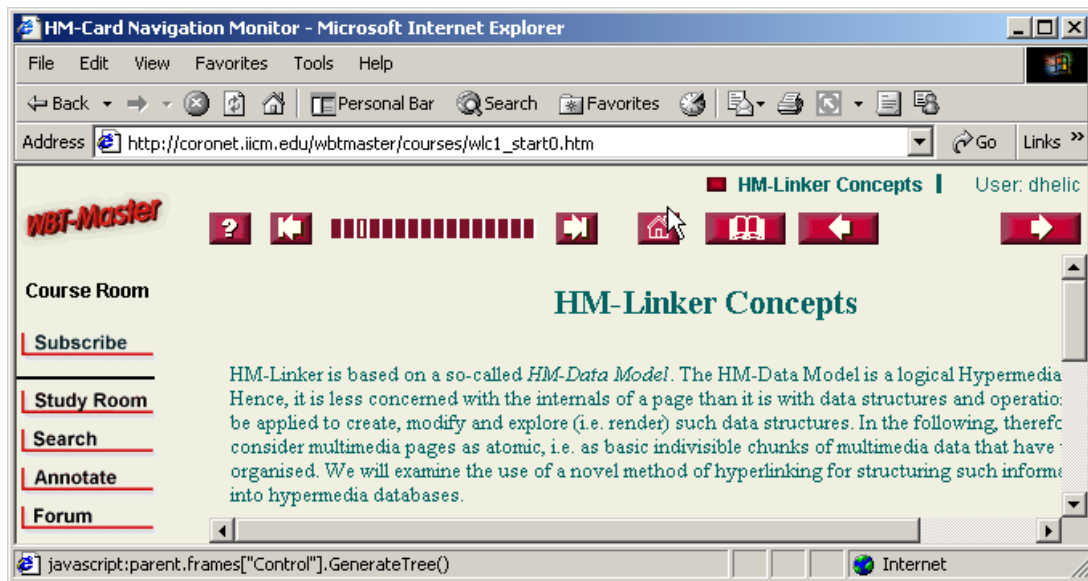


Figure 4.3 WBT-Master functional panel

Each functional panel provides an interface to a particular data object. Thus, we can speak about “Course” panel, “Forum” panel, “Learning Goal” panel, etc.

Since WBT-Master supports a big variety of actions, which can be applied to a single data object, a main functional panel may invoke other functional panels performing actions on the same data object.

Thus, for example, there may be a “Course Browser” panel, “Course Administration” panel, “Course Annotation” panel, etc.

4.2 HC-Data model in WBT-Master

4.2.1 Working with HM-Linker

HC-Units may be created manually as text files in accordance with the predefined format or by means of a special authoring component called HM-Linker [WBT-Master, 2001]. HM-Linker is based on the presented HC-Data model.

HM-Linker allows HC-Units to be quickly assembled, given existing materials (i.e. HTML and other WWW compatible documents).

HM-Linker control panel consists of three main elements:

- Embedded file browser (a)
- S-Collection template (b, c, d and e)
- Preview area (f).

The HC-Type template, in turn, consists of a number of cells, which control the following attributes of an HC-Unit:

- The cell (b), which, in turn is divided into three cells (c) defines documents or other HC-Units, which were selected as Label, Head and Members.
- Cell (d) define parameters for a selected HC-Unit element.
- Cell (e) define global parameters for the current HC-Unit (HC-Unit, type, name and location).

Creating a new HC- Unit:

- Click on one of “New Composite” icons to create a new HC-Unit of the selected type. Once a new HC-Unit has been created, parameters can be modified, and members can be inserted, modified or removed.

Selecting an HC-Unit:

- Use the embedded file browser to select one of existing HC-Units (files with extension *.cif). Once an existing HC-Unit has been selected, parameters can be modified, and members can be inserted, modified or removed.

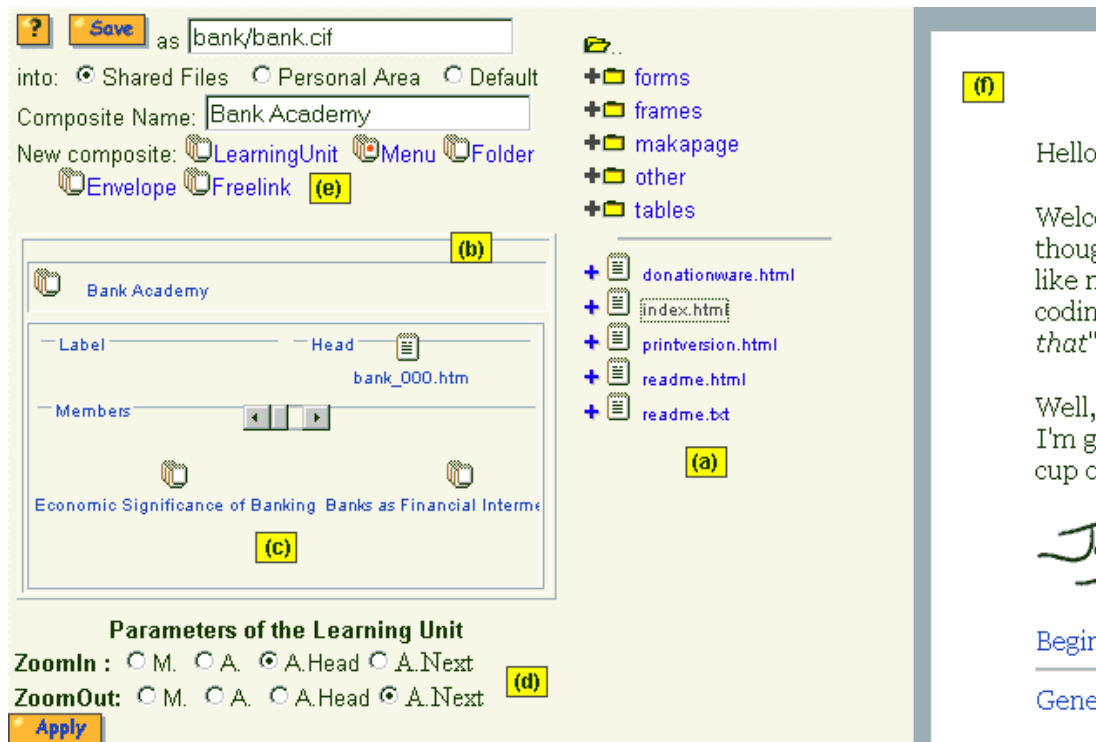


Figure 4.4 HM-Linker

Setting global HC-Unit parameters:

- Type in a desired Composite Name;
- Click on the composite icon in the upper left corner of the cell (b) and select a desired combination of Zoom-In and Zoom-Out parameters;
- Confirm your parameter selection with the “Apply” button;

Adding a new member to the HC-Unit:

- Select a cell where the new member should be placed by clicking on it; If the cell contains previously defined members, click on one of such members to define a position where the new member should be inserted (just before the selected member). If no members were selected, the new member will be inserted as a “last” member of the cell.
- Select a desired member (document or another composite) using the embedded file browser;
- Click on the “+” symbol to add the member. If you need to change a position of existing member within the composite cell, simply drag & drop it to a desired position within the composite cell.

Removing an existing member from the HC-Unit:

- Click a designated member with right mouse button, a special pop-up menu appears;
- Select “Delete” entry from the pop-up menu.

Setting parameters for an existing member.

- Click a designated member with left mouse button, the member icon will be marked with a red dot and a special parameter form appears in the cell (d);
- Set a desired combination of parameters.

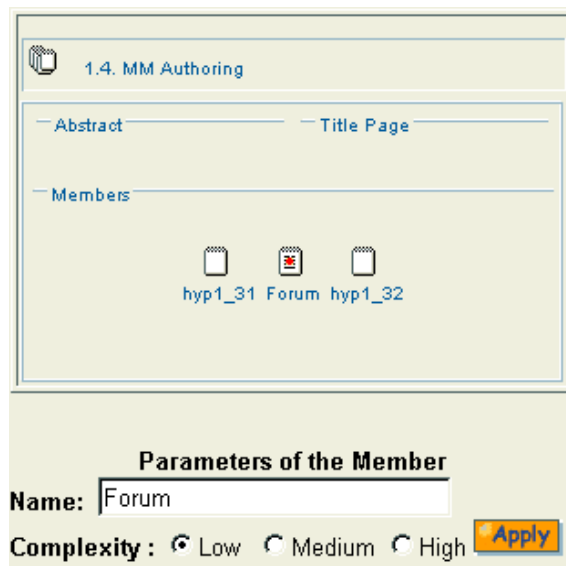


Figure 4.5 Setting member parameters with HM-Linker

Note, the member name is essential since it is used to display the document in the course map and the table of contents. The complexity parameter should be used if the member encapsulates some non-HTML elements or uses frames. Thus, for example, the complexity parameter should be set to “medium” if the document embeds Complex Java Script elements, Java applets and/or references to a Plug-In. If the document just sets a frame structure for other documents, the complexity parameter should be set to “high”:

- Confirm your parameter settings with the “Apply” button;

Converting a server directory into a composite.

An existing server directory may be treated as a composite using the following rules:

- All the files of chosen type (say all HTML documents) are considered to be the composite members;
- All the members are sorted alphabetically by the file name;
- All subdirectory are considered to be members of the composite and the rules are recursively applied to such subdirectories.

To convert a directory into a composite:

- Click a symbol (+) just before the directory name with left mouse button, the system prompt user to list all the file types to be inserted as the composite members.
- Type in all chosen file types extensions using ‘,’ as a delimiter
- Confirm your intention to convert the directory with the “OK” button.

Putting a resultant composite on the server:

To store the resultant composite on the server:

- Select a repository where the composite should be stored (Shared Files, Personal Area or Default).
- Type in a desired composite title and the composite file name (note the composites are stored on the server as files with the extension *.cif)
- Click on the “Save” button.

4.2.2 HC-Units Navigation Panel

Course Navigation Panel provides a number of functional buttons to control the flow of navigation in a particular course. This panel invokes other functional panels, such as Course Location Feedback Panel or Course Annotation Panel, which perform additional actions on the same course data object.

Note that a course in the WBT-Master is just an instance of a particular HC-Type, i.e., an HC-Unit.

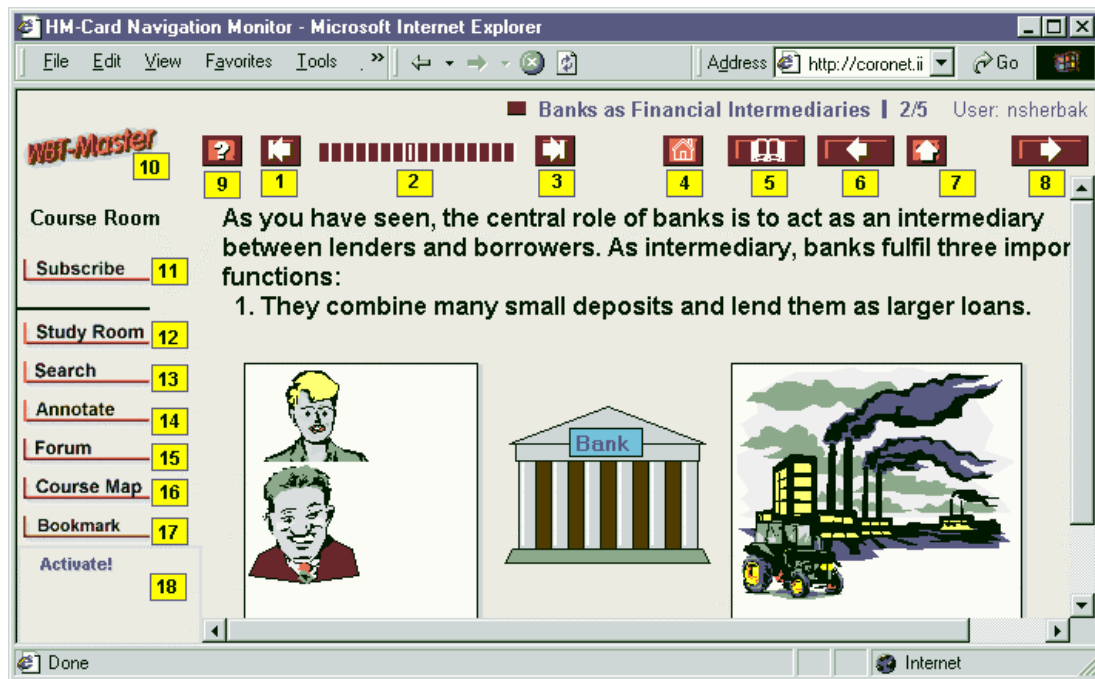


Figure 4.6 Course Navigation Panel

Here is the description of the enumerated functional buttons shown in the Figure 4.7:

1 - *Back to the previous document*

This button alters the current user location in a particular course: it gets the user to the previous member of the course data object. If the current user is subscribed his/her current position is always tracked on the server.

2 - *Activating the Course Map window*

The Course Map window shows an overview of a particular course and its members. The whole course hierarchy is presented to users in the Course Map.

It should be noted that the Course Map provides also an access to an arbitrary position within the course by just clicking on that position.

3 - *Jump to the next document*

This button leads users to the next member of a particular course.

4 - *Getting list of all courses*

Activating this button leads users to Course Selection Panel. Course Selection Panel provides a list of all courses on the server.

5 - *Back to the table of contents*

Applying this action to a particular course results in visualization of the course entry document. The entry document is a combination of the title and the abstract course page along with a course hierarchy overview. In the hierarchy overview users current location in the course is visualized, thus a location feedback mechanism is provided.

6 - *Back to the previous execution step*

This button provides facilities for navigating in a new dimension, namely the time dimension. If the current document is an animation or other time-based document activating this button leads to the execution of the previous animation step. If no step was accomplished yet then the previous course member is visualized.

7 - *Open/Close current composite*

These buttons provide an alternative way of browsing using “Zoom In” and “Zoom Out” operations.

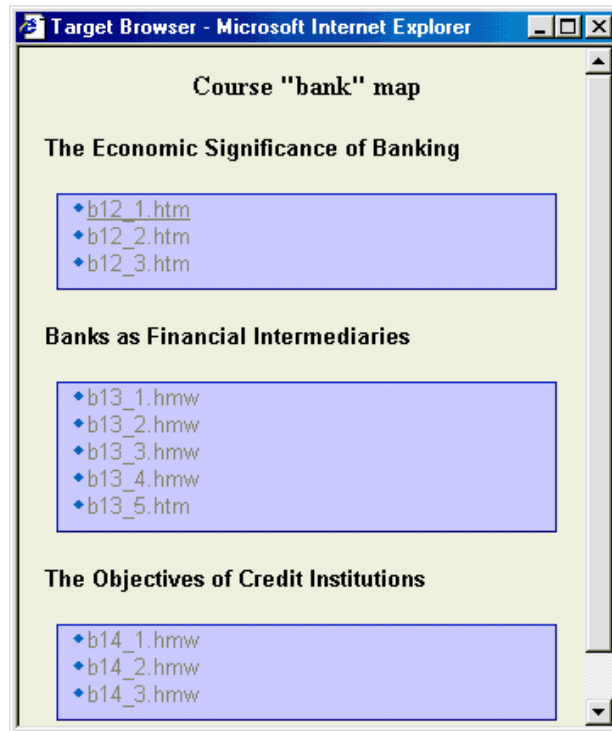


Figure 4.7 Course Map

8 - Jump to the next execution step

This button activates the next execution step of a time-based document. If the document is not time-based or all time-based steps were already accomplished then activating this button leads to visualization of the next course member.

Just above the navigational tools you will find the title of the current chapter, number of the document within the chapter and information on the user's registration.

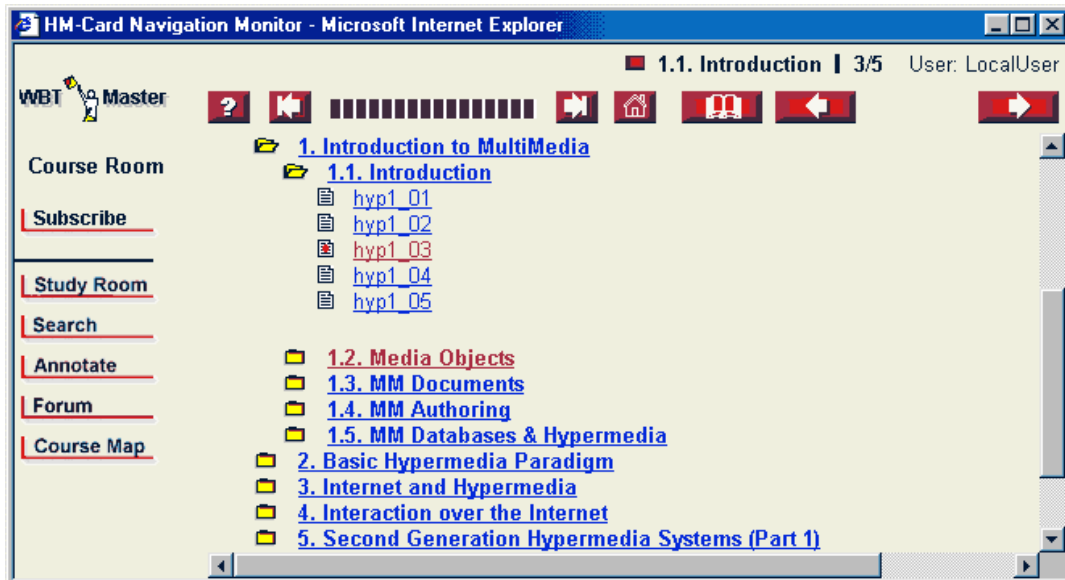


Figure 4.8 Course Table of Content

9 - Entering course help system

This button opens the course help system in a new window. The course help system provides a hypermedia description of Course Panels as shown in below.

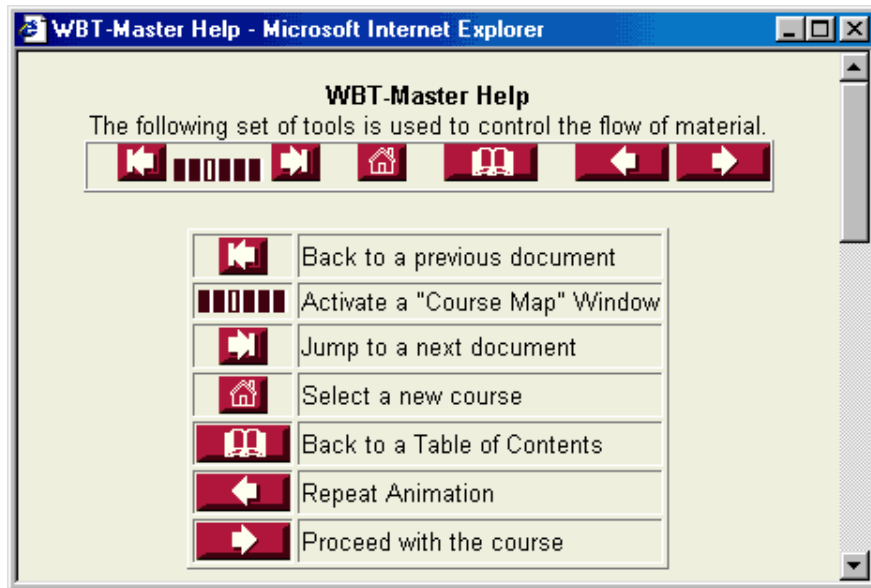


Figure 4.9 Help Window

10 - Getting list of all courses

Activating this button leads users to Course Selection Panel. Course Selection Panel provides a list of all courses on the server.

11 – Subscribe

Learners can use this button to subscribe for the course. Subscribed users get additional rights for the course content.

12 - Getting Study Room functional panel

This button returns you to the Study Room where a particular course and viewer (look & feel) may be selected.

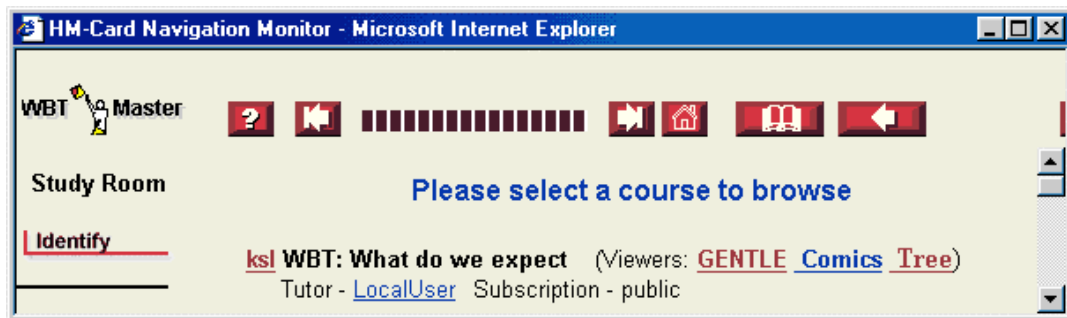


Figure 4.10 Selecting "Look and Feel" for a particular course

13 - Getting Course Search functional panel

Users can use this button to search for certain topics within a course. If this button is activated the Course Search Functional Panel will be opened. It provides facilities for a context-dependent full-text search.

14 - Getting Course Annotation functional panel

This button enables users to add notes to the course content. If users click on this button the Course Annotation Functional Panel will appear. It provides a possibility to write a course specific annotation.

15 - Entering course specific discussion area

This button enables users to enter the course specific discussion area and write articles pertaining to the course.

16 - Activating the Course Map window

This button provides an overview of the course structure along with a direct access to individual documents.

17- Bookmark a current document

This button adds a current document to the user's personal bookmarks. To add the document to the bookmarks: click on the button (17), a special "Enter the document title" window appears. Type a desired title for the document and press "OK".

18- Existing Bookmarks window

This window displays all the personal bookmarks referring to documents belonging to the current course. In order to activate the bookmarks window (i.e., to get all the bookmarks from the server) click on the "activate" symbol. To jump to a bookmark positions simply click on the document title.

4.3 Knowledge Domains in WBT-Master

Knowledge Domains in WBT-Master are supposed to provide WBT-Master users with a general semantic overview of all learning resources in the system, i.e., with knowledge profiles. Thus, each particular knowledge domain is a collection of WBT-Master training resources (documents, learning units, individual users, etc.), which are structured according to the corresponding knowledge domain schema.

The WBT-Master GUI provides a number of functional panels that allow users to work with knowledge domains. This concept is supported by means of three WBT-Master tools [WBT-Master, 2001]:

- Knowledge Domain Schema authoring tool
- Knowledge Domain Content administration tool
- Knowledge Domain Content browser

The following few sections provide the description of those tools.

4.3.1 Working with Knowledge Domain Schema authoring tool

Whenever a user activates the schema-authoring tool, the following functional panel is displayed.

The panel provides help function (1), entry to the knowledge domain browser (2) and entry to the knowledge domain content administration system (3).

The main functionality is accomplished via the buttons "Edit an existing schema" (4), delete an existing knowledge domain (5) and create a new knowledge domain (6).

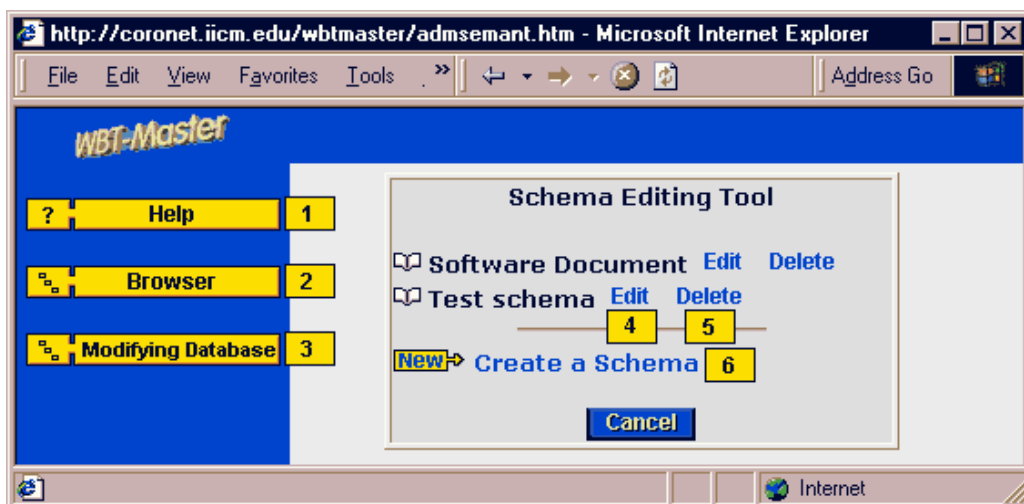


Figure 4.11 Knowledge Domain Schema Functional Panel.

To create a new knowledge domain, click the button “New” (6), and fill out the following form describing the knowledge domain for users:

Figure 4.12 Defining a new Knowledge Domain Schema

After defining the general attributes for a new Knowledge domain or selecting an existing schema for editing, the user can define semantic categories and relationships between them.

Figure 4.13 Defining categories and relationships

To define a new category click on the button “Create a Category”, fill out the following form and click “Save”.

Figure 4.14 Defining a new semantic category

A new category without any additional attributes will be created and added to the schema. Repeat the process to define all the categories needed to accomplish your application.



Figure 4.15 Schema with defined categories

Use “Delete” button (2) to delete a category and “Edit” button to add/edit attributes. To add a new attribute to a category, click on the “Edit” button (1), the “New Item” button and fill the following form out.

Figure 4.16 Defining a category item

For each attribute, you are supposed to provide the attribute name and to set a particular type of values (may be: String, Numeric, Text or Selection). For “String” and “Numeric” an approximate number of symbols in each value is needed. This value is used to create an appropriate input field for entering the attribute value. For “Selection”, list all possible values using “/” as a delimiter. For example, an attribute “Programming language” might be defined as a “Selection” with the following predefined set of values “Java/JavaScript/C/C++” if no other languages is used in the organization.

Repeat the process to define all the attributes needed for your application.

To add a new relationship to the schema, click on the “Create Relationship” button (3) and fill the following form out.

Current version of WBT-Master has the following restrictions for creating relationships between instances of categories.

Each relationship define 1:n relations between instances of two different categories

A category participating with a single instance in each relationship is called “Owner” of the relationship.

A category participating with multiple instances in each relationship is called “Member” of the relationship

Each instance of a Member is obligatory related to a certain instance of Owner, this reference is installed whenever a new Member is created.

Define new Relationship

Title for Owner: Courses

Owner (1) : Trainer

Title for Member: Trainer

Member (n) : Training Course

Relationship is: Mandatory Optional

Save Cancel

Figure 4.17 Defining a new relationship

For example, the relationship defined above may relate a single instance of Trainer to an arbitrary number of Courses, or looking from another perspective, each instance of Course is obligatory related to a certain Trainer. This relationship will be created whenever a new Course instance is put into the Knowledge domain.

The fields “Title for owner” and “Title for member” are used to visualize the relationship during browsing.

Composite "Scheduled Training Sessions"

- Trainer Edit Delete
- Training service customer Edit Delete
- Training Course Edit Delete

New → Create a Category

- Courses (Trainer → Training Course) Delete
- Courses (Training service customer → Training Course) Delete

New → Create a Relationship

Cancel Preview

Figure 4.18 The complete schema

Thus, if a user accesses a member instance, it will be visualized along with a reference to an owner entitled as it set by the attribute “Title for member”. Similarly, if a user accesses an owner instance, it will be visualized along with references to all members entitled as it set by the attribute “Title for owner”.

Repeat the process to define all the relationships needed for your application. Finally, the schema may look as in figure 4.19. The schema may be also previewed as a graph using the “Preview” option :

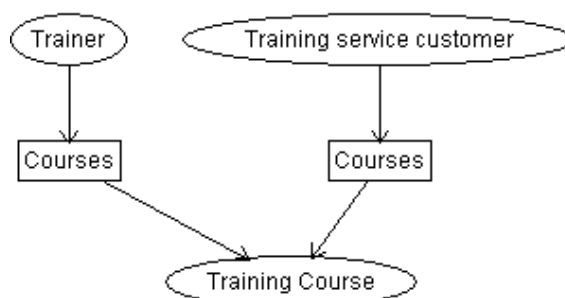


Figure 4.19 Previewing the defined schema

4.3.2 Working with Knowledge Domain Content administration tool

Whenever a user activates the knowledge domain content administration tool, the following functional panel is displayed.

The panel provides a help function (1), entry to the knowledge domain browser (2), entry to the server content administration panel (3), and knowledge domain schema definition tool (4).

The main functionality is accomplished as copying training resources from the server (5) to a particular knowledge domain (6) as instances of predefined categories.

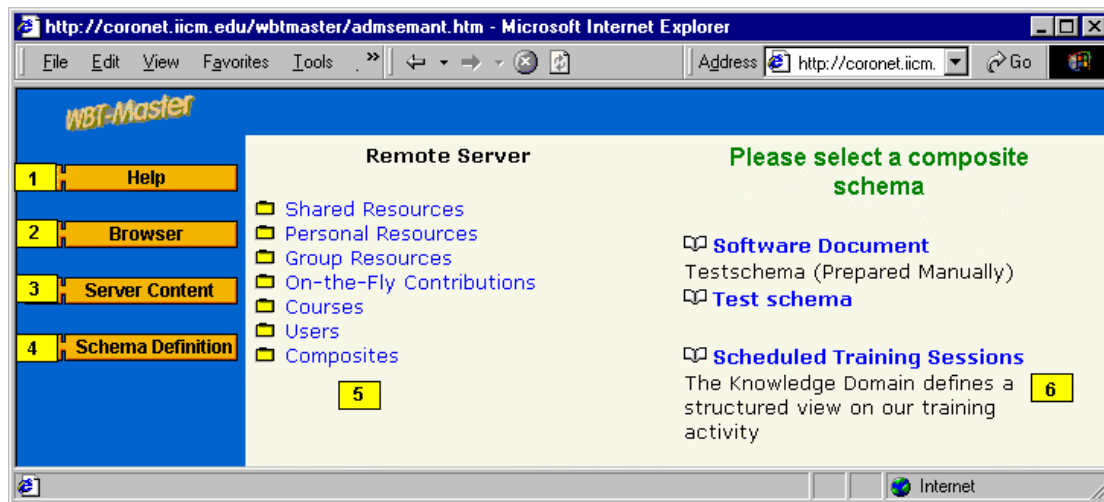


Figure 4.20 Knowledge Domain Content Administration Functional Panel

To create a new instance of a particular category, select a previously defined knowledge domain (6), and the category. Browse the remote server content (5) to locate a designated training resource and click the symbol (+) in front of the selected resource name.

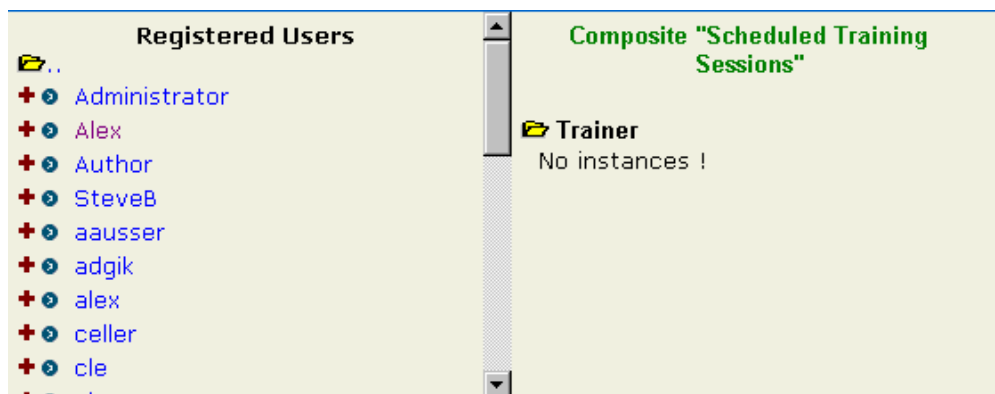


Figure 4.21 Adding a training resource as a category instance

The form appearing in the left part of the screen very much depends on the category definition in the selected schema. It prompts to input all the attributes defined for this category and to select instances of other categories references to which were predefined.

Thus, for example, creating a new course instance may result in the following form where the attribute Title was defined for the category course and references to Trainer and to Customer were defined as relationships.

After setting all the necessary values click on the “Save” button to actually create the new instance and add it to the knowledge domain. Repeat the process to add more category instances to the knowledge domain.

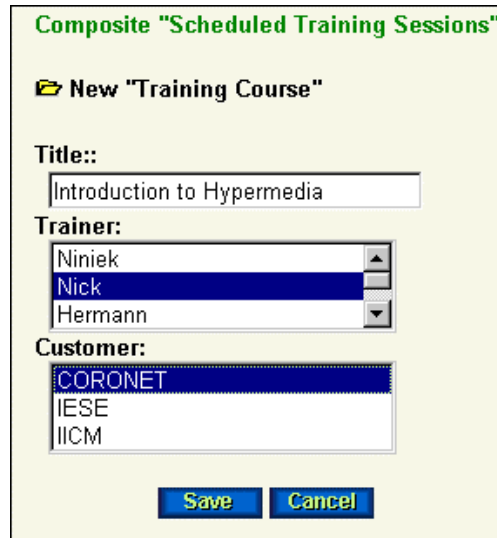


Figure 4.22 Defining attributes and installing relationships for a new category instance

Needless to say, new instances may be added to the knowledge domain later at any time.

4.3.3 Browsing a Knowledge Domain

Browsing a knowledge domain does not require any additional knowledge besides understanding of the predefined structure of the domain. The user simply select a particular knowledge domain and a certain category (1), all existing instances are displayed (2). The “Cancel button can be always used to return to a previous navigational step.

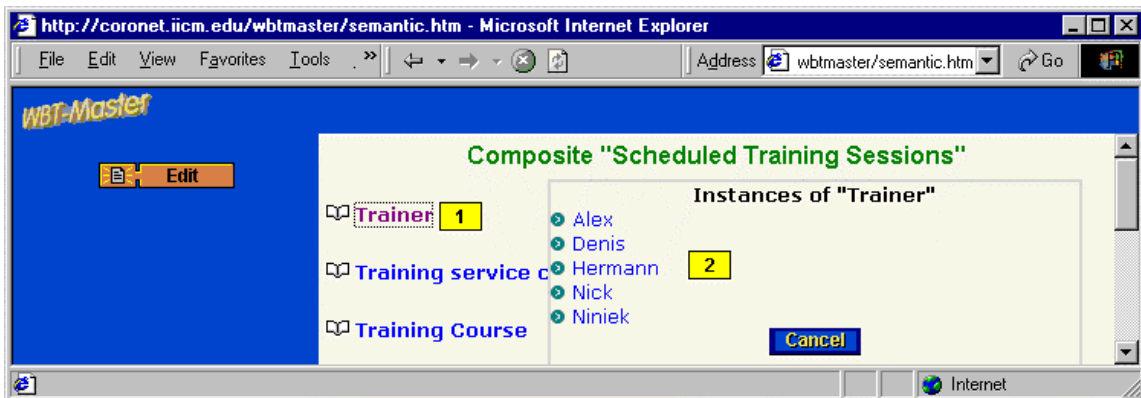


Figure 4.23 Accessing all instances of a particular category

Selecting a particular instance with a mouse click, results in displaying all the instance attributes and references to other instances.

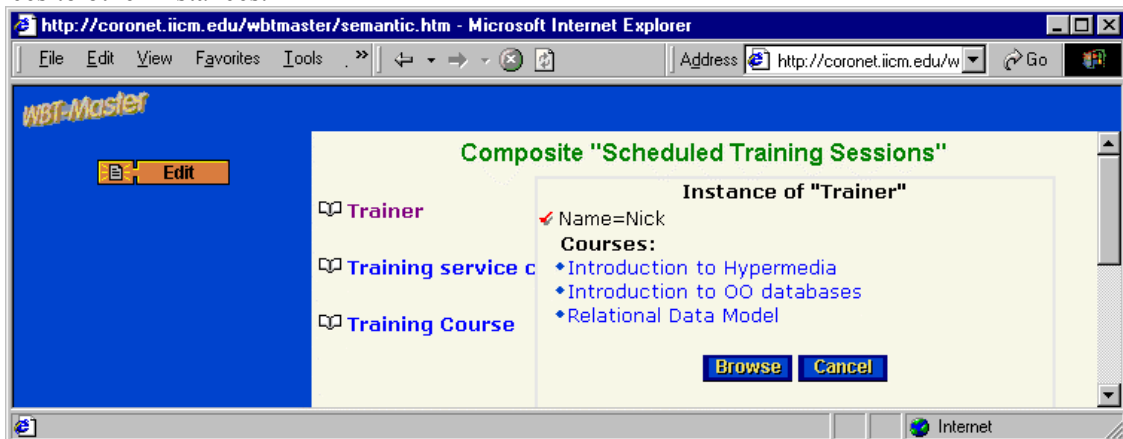


Figure 4.24 Previewing a category instance

Now, the user may select another instance by clicking on its name or start actual browsing of the knowledge domain by clicking on the “Browse” button.

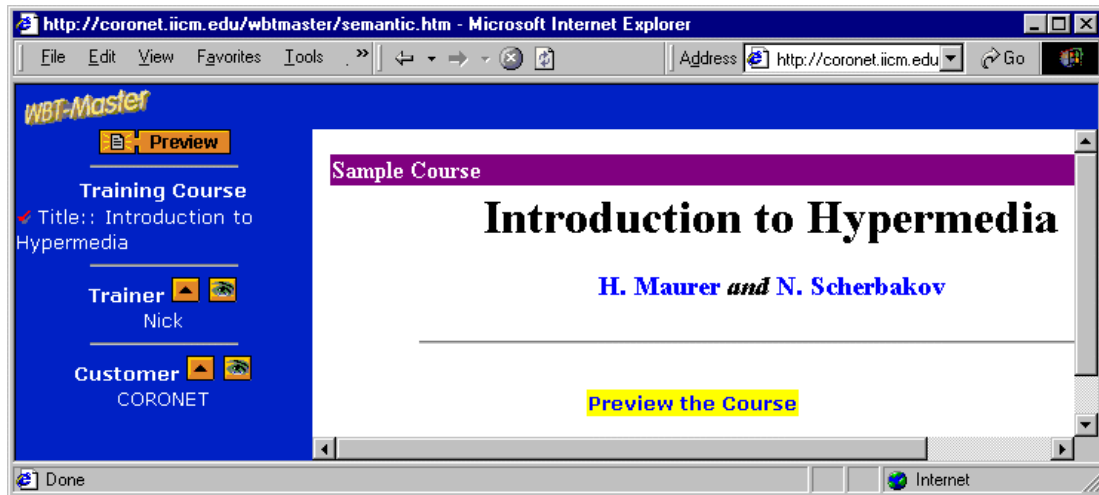


Figure 4.25 Browsing a category instance

Navigational tools on the left part of the screen may be used to access or preview instances of other semantic categories related to the current one.

4.4 Knowledge Cards in WBT-Master

Knowledge Cards are supported in the WBT-Master by a number of so-called Knowledge Card functional panels.

In the next few sections I describe all Knowledge Card functional panels including Knowledge Card Control Panel and Knowledge Card Administration panel [WBT-Master, 2001].

4.4.1 Knowledge Card Control Panel

The Knowledge Card Control Panel can be seen as a main entry point to the system for users looking for Learning Resources.

A particular knowledge card may be selected by clicking on its title. Such selected Knowledge Card becomes a so-called current card (3). Resources associated with a current card (3) may be requested by clicking on the “Get Know Card” icon (4). The Knowledge Card (5) provides a convenient way of accessing the learning resources by clicking on its title.

Here is the description of the enumerated functional buttons shown in the Figure 4.27.

1 - *Getting list of all courses*

This button activates the list of all courses on the server, i.e. Course Selection Panel is visualized.

2 - *Entering Knowledge Card Administration System*

The Knowledge Card Administration System allows to create new Knowledge Cards and to modify parameters of existing Knowledge Cards.

3 - *Accessing Knowledge Cards related as “is-a-part-of” to the current one.*

If a Knowledge Card is visualized as a “folder” (■), then a number of other Knowledge Cards refer to this one as “is-a-part-of”. Clicking on the “folder” symbol provides access to such referencing cards where the process may be repeated.

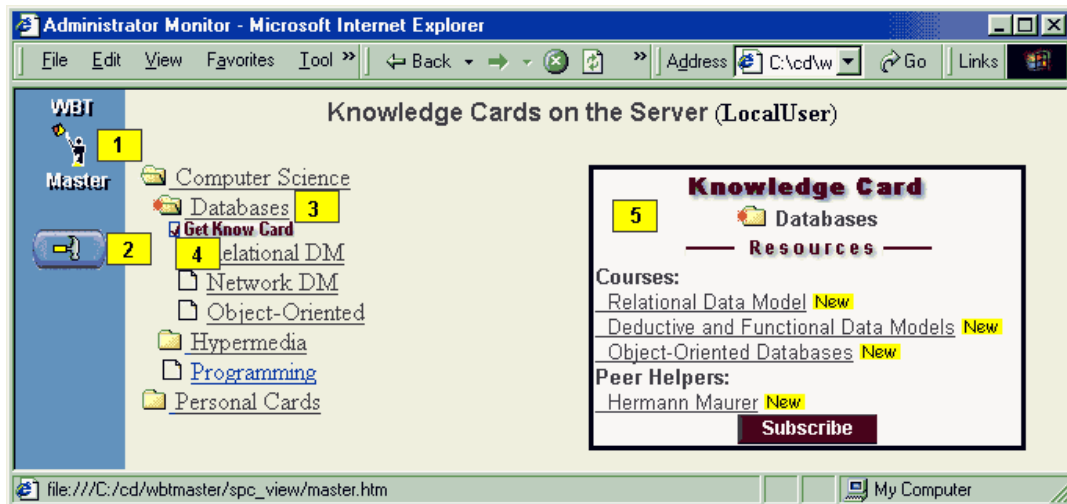


Figure 4.26 Knowledge Card Control Panel

4 - Visualizing the Knowledge Card.

Resources associated with a Current Card (3) are requested by clicking on the “Get Know Card” icon. All the resources associated with the current card are inferred after this explicit request.

4.4.2 Knowledge Card Administration Panel

The Knowledge Card Administration System allows to create new Knowledge Cards and to modify attributes of existing Knowledge Cards.

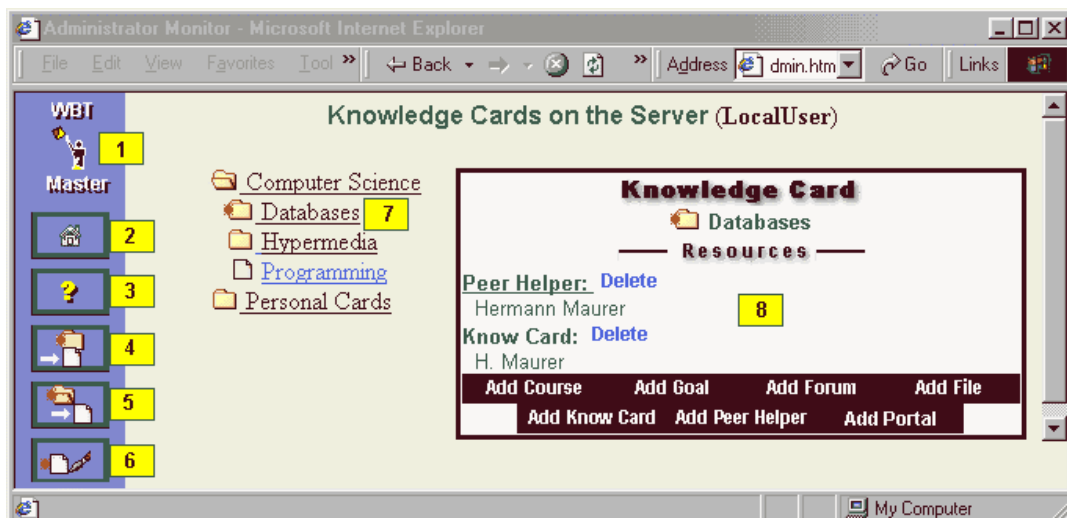


Figure 4.27 Knowledge Card Administration Panel

Here is the description of the enumerated functional buttons shown in the Figure 8-2.

1,2 - Getting Knowledge Card Control Panel

Clicking on this button leads users back to the Knowledge Card Control Panel.

3 - Entering Knowledge Card Administration System Help

Opens a new window containing a hypermedia description of the Knowledge Card Administration System.

4, 5 - Creating a new Knowledge Card

The button provides a new input form for defining attributes of a Knowledge Card. The following parameters should be defined:

- Knowledge Card Title: text defining the Knowledge Card itself.
- Concept Definition: text defining the concept, which is supposed to be presented by resources associated with the Knowledge Card.

The Apply button stores the new Knowledge Card on the server.

The newly created Knowledge Card can be related as “is-a-part-of” to a Current Card (5) or to the same concept as the Current Card (4). In other words, the new Knowledge Card can be inserted on the same level as the Current Card (4) or to the list of Knowledge Cards defining sub-concepts for the Current Card (5).

If no Current Card was previously selected, the new Knowledge Card is not related to existing Knowledge Cards, i.e. it is inserted on the first level.

6 – Modification of an existing Knowledge Card

Attributes of a Current Card (7) may be modified by means of this button.

8 - Adding a new learning resource

When a Current Card is selected, it is visualized with a list of resources attached to this particular Knowledge Card. A resource may be deleted from the list of resources by clicking the “Delete” icon. To add a new resource to the list, one of the buttons below (“Add Course”, “Add Goal”, “Add Forum”, “Add File”, “Add Peer Helper” or “Add Portal”) should be used.

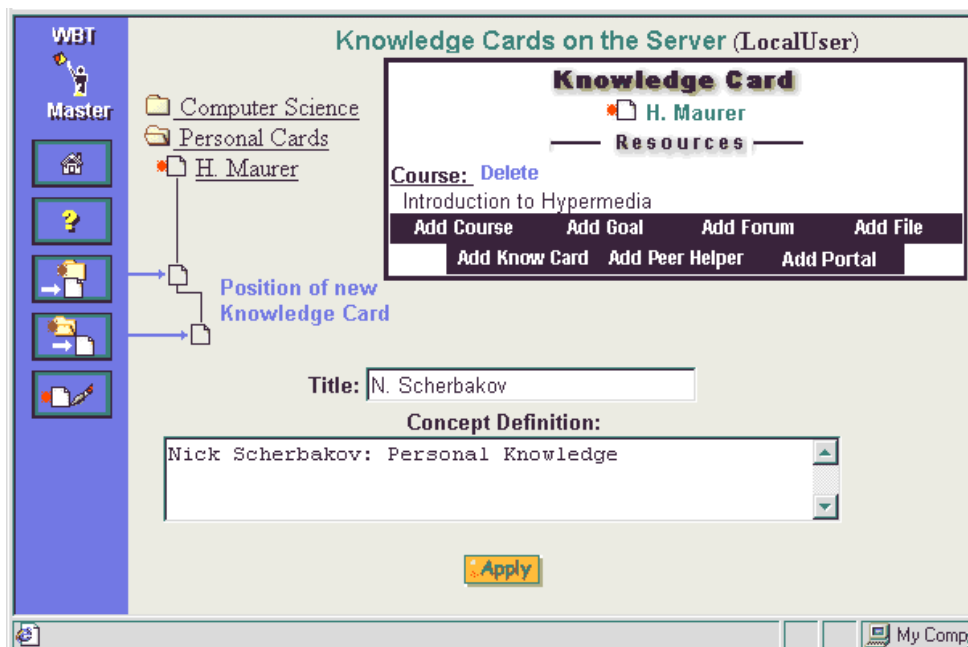


Figure 4.28 Defining a new Knowledge Card

Clicking on one of this buttons results in opening a resource browser that allows selecting a particular learning resource available from the server. The selected resource is automatically added to the list of resources attached to the Current Card.

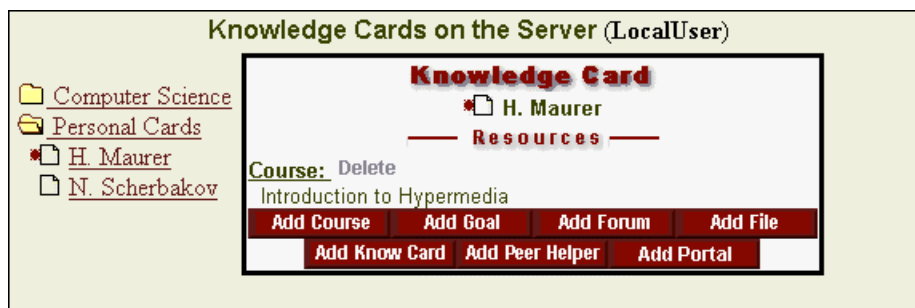


Figure 4.29 Adding a Learning Resource

5 Related Work

This chapter provides the overview of the related work. I tried to investigate data modeling approaches that utilize similar concepts in the terms of semantic data modeling for hypermedia systems. Thus, I did not pay much attention to implementation issues of any particular hypermedia system, but rather I put my attention to semantic data modeling concepts behind these implementations.

I start this overview with a short description of systems utilizing some aspects of semantic data modeling in hypermedia systems, but which are in my opinion by no means advanced as semantic data modeling concepts that I presented in this thesis. For each of these concepts I provide a short list of disadvantages of such approaches in respect to the presented approaches.

Finally, I introduce two semantic data modeling initiatives, namely Resource-description-framework (RDF) and Topic-map initiatives, which allows for far more sophisticated modeling of a hyperweb. I conclude each of these introduction with an in-dept analysis of similarities and differences of these approaches to the presented approaches. Also illustrative examples showing possible mapping between these approaches are given.

5.1 Hypermedia systems utilizing semantic data modeling

The explicit representation and use of semantic knowledge about a domain to facilitate or guide the access to information has been a primary concern in hypermedia systems from early times [Al-Khatib et al., 1999; Gains and Shaw, 1995; Castells and Szekely, 1999]. Most of them are based on the level of content that is discrete on the basis of some kind of elementary unit, and the level of semantic structure, that is used to road map to guide navigation. There is a great variation of how content is structured, how the semantic structure is organized and how both of such levels are connected together in the system [Castells and Szekelly, 1999]. Here I provided a short list of such systems.

5.1.1 Interbook

Interbook [Brusilowski et al., 1998] structures elementary units (e.g. courses) into hierarchical aggregate units: chapters, sections, subsections and terminal pages. These units are accompanied by a set of interrelated concepts with two possible relation types: prerequisite and outcome. Thus, Interbook provides a relative simple two-relation model, which contrasts with the lexical richness of Knowledge Domains for instance. Moreover, users can just navigate the hierarchical structure offered by the system, and there is no possibility to manipulate in any way.

5.1.2 HyperTutor

In the contrast to Interbook, HyperTutor [Perez et al., 1995] provides very rich lexical structures allowing creating conceptual maps with a wide variety of relations from the educational theory: prerequisite, sequence, aggregation, similarity, opposite, example, specialization and exception. However, these relations

are predefined and there is no possibility offered by the system to define a new and possible different set of relations. Recollect that such possibility is an ingredient part of Knowledge Domain concept.

5.1.3 PEGASUS

PEGASUS stands for Presentation Modeling Environment for Generic Adaptive Hypermedia Support System. PEGASUS is a generic hypermedia presentation system that makes minimum assumptions about how instructional knowledge is represented [Castells and Szekely, 1999]. PEGASUS provides hypermedia designers with a simple specification paradigm for non-trivial adaptive presentation constructs. To allow different approaches PEGASUS supports the definition of domain ontology for the description of terms and their relationships for a particular domain. Once ontology is created authors simply assign hypermedia objects to the defined terms and relate them according to the defined relationships. However, PEGASUS is mainly presentation system putting the main effort in providing a powerful presentation model. Such presentation model supports an application of so-called presentation templates.

5.1.4 Ontobroker

Ontobroker [Fensel et al., 1998; Fensel et al., 1998a; Decker et al., 1999], which uses formal ontologies to extract, reason, and generate metadata in the WWW is a system developed at Institute for Applied Computer Science and Formal Description Methods (AIFB), Universität Karlsruhe (TH), Karlsruhe, Germany. The former name of the system was On2Broker. The system works with the formalisms and tools for formulating queries, defining ontologies, extracting metadata, and generating metadata in the format of the Resource-description-framework (RDF) for the Web [Fensel et al., 1998]. These methods provide a means for semantic based query handling even if the information is spread over several sources. Furthermore, the generation of RDF descriptions enables the exploitation of the ontological information in RDF-based applications. Recently, Ontobroker was further developed to become SEAL and SEAL-II, so-called knowledge portals for the Web. These two systems include powerful visualization mechanisms for underlying knowledge structures [Hotho et al., 2001].

5.2 Resource Description Framework

RDF stands for Resource Description Framework. The main goal of the Resource-description-framework initiative is to provide an integration platform for a wide range of heterogeneous Web applications. To do so RDF tries to insure interoperability at the level of meta-data specifications of different applications [Corby et al., 2000; Calvanese et al., 1998a]. That means that different Web applications supporting RDF specifications and schemas might process transparently a variety of web-based meta-data specifications including sitemaps, content ratings, stream channel definitions, search engine data collections (web crawling), digital library collections, and distributed authoring.

Thus, the RDF specifications provide a lightweight ontology system to support the exchange of knowledge on the Web [Berners-Lee and Fischetti, 1999; Berners-Lee et al., 2001]. As the data exchange format RDF uses XML syntax.

RDF started as a W3C (Web Consortium) activity that eventually evolved into a so-called Semantic Web activity. The W3C Semantic Web Activity Statement [Semantic Web, 2001] explains W3C's plans for RDF and metadata in detail.

5.2.1 The Semantic Web

The Semantic Web technology may be summarized as follows:

- RDF [RDF, 2001; Lassila and Swick, 1999] is applied to express meaning of resources residing on the Web to the computer programs (software agents) that come across these resources. The basic RDF model encodes meanings in the form of set of triples, each triple being like a subject, a verb and an object of an elementary sentence. These triples are written using XML tags. Thus, in RDF a

document is a set of assertions that a particular resource on the Web (Web page, the person owning a Web page, etc.) has property (“is the homepage of”, “is employed by”, etc.) with a certain value (another Web page, an institution, etc.). An URL identifies subject, object and even verb. Having the verb being identified by a URL allows for defining properties of properties, i.e., meta-properties. These might be used to insure interoperability between different meta-data initiatives. For instance, a property of a particular property might define its synonym in another meta-data specification.

- Structured knowledge represented by means of RDF documents may be accompanied with so-called inference rules. Software agents in order to automatically conduct reasoning and derive new facts may use such inference rules. For instance, new facts might be descriptions of a particular resource in terms of another meta-data specification. The language for inference rules is made as expressible as possible to allow the Web to reason as widely as desired [Berners-Lee et al., 2001]. However, the price for this expressiveness is that software agents will have to face a paradox or an unanswerable question now and then. This approach is similar to that of the conventional Web development. The Web does not ensure the consistency of its entire links, but allows for a simple system that has an exponential growth. Similarly, the Semantic Web does not guarantee that there want be a paradox in the system, but again it allows for a very simple system that probably would have (again) an exponential growth.
- Semantic Web supports the concept of ontology, which is a collection of information that describes objects (concepts) and relationships between them. Because there may be many different ontology definitions describing one and the same domain, ontology might also include a set of inference rules specifying how this particular ontology relates to others, by specifying for instance equivalence relations between terms from the original and other ontology definitions.
- Software agents are programs that collect Web content from diverse sources, process the information and exchange the results with other programs (agents) [Berners-Lee et al., 2001]. An important facility of agents’ functioning will be exchange of “proofs”, where an agent that collect some information will ask another agent to confirm, i.e., “prove” that the collected information is indeed the needed (desired) information. Another rather important feature of agents’ functioning will be that of exchanging digital signatures between different agents, that will allow one agent to trust the other one. All that being said, Semantic Web is supposed to have a vast number of different agents working together to answer a particular question. A typical process will involve the creation of a "value chain" in which subassemblies of information are passed from one agent to another, each one "adding value," to construct the final product requested by the end user [Berners-Lee et al., 2001].

Thus, one of the main concepts of the Semantic Web is to provide a unifying semantic knowledge representation of heterogeneous Web of today. The RDF is used to provide the vast resources of the World Wide Web (and beyond) with a set of meta-data that is understandable by computer programs (software agents). Those agents will be responsible for inferring relevant Web resources upon a user’s request.

5.2.2 Knowledge Domains on the Semantic Web

Knowledge Domains apply semantic modeling of hypermedia data to provide a set of hypermedia resources with a general navigable and searchable overview, i.e., to profile knowledge inherent in this set of hypermedia resources. Such semantic modeling is based on the use of ontology to identify the terms and relationships between those terms. Hypermedia resources from the underlying set are assigned to the previously defined terms and related by means of their interrelationships. In this way, concept and relationships between concepts (i.e., knowledge) that are implicit to the set of hypermedia resources are made explicit and available for users to browse or search through it.

Thus, similarities between the Semantic Web and Knowledge Domain are obvious:

- Both apply ontology definitions to define terms and relationships between those terms
- Both assign hypermedia resources to the terms and relate them by means of the defined relationships

However, we may identify a number of differences as well:

- Knowledge Domains are mainly used to provide a navigable, searchable semantic overview of the resources, i.e., to profile knowledge contained in these resources; The Semantic Web, on the other

hand, provides facilities to switch back and forth between a number of such profiled knowledge representations, thus making these representations transparent to their end-consumers.

- The Semantic Web is far more general because it might be used to define ontology but also meta-data specifications, transformation specifications between two or more different meta-data specifications, etc.
- Knowledge Domains are much more hypermedia-specific and hypermedia-application-oriented. Usually, as much expressiveness as RDF offers is not needed to support a particular Web-oriented hypermedia application and might require an additional work load on the side of authors/administrators/knowledge engineers.

All said, the conclusion here is that Knowledge Domains might be seen as a perfect tool for authoring/browsing/searching structured knowledge library of hypermedia resources. Moreover, this tool may be easily made compatible with the Semantic Web. For instance, by mapping a particular Knowledge Domain ontology, i.e., a Knowledge Domain Schema to an RDF Schema that schema gains the full operability by means of the Semantic Web.

Additionally, one might imagine defining a number of RDF specifications that describe mapping of that particular ontology onto other ones residing somewhere on the Semantic Web. This would mean interoperability at the level of different ontology specifications.

Finally, dumping content of Knowledge Domain into RDF documents, i.e., dumping resource assignments to RDF statements, would make that particular Knowledge Domain fully Semantic Web compatible and interoperable with other descriptions residing on the Semantic Web.

5.2.3 Example of mapping Knowledge Domain onto RDF

Let us consider the example that we had before. The document "C" is a technical description of the software module implemented by the programmer "A" for the project "B".

We identified the three different semantic categories here:

- "Module" (with name and programming language as data items)
- "Project" (with name as a data item)
- "Author" (with name as a data item)

The semantic relationships between these categories were defined as follows:

- "Project Modules" 1:n relationship between "Project" and "Module" category
- "Author Modules" 1:n relationship between "Author" and "Module" category.

The above-mentioned example may be instantiated as follows: "C" (written in Java programming language) is an instance of "Module" category, associated with the "Project" "B" and written by the "Author" "A".

This example written in RDF would include two RDF documents:

- Document defining the RDF schema
- RDF document describing the resources according to the RDF schema

Here is an example of a RDF schema introducing the terms from the above example:

```
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<--
  Categories
-->
<rdf:Description ID="Project">
```

```

<rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<rdf:Description ID="Module">
<rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<rdf:Description ID="Author">
<rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<--
  Relationships
-->

<rdf:Description ID="name">
<rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
<rdfs:domain rdf:resource="#Project"/>
<rdfs:domain rdf:resource="#Module"/>
<rdfs:domain rdf:resource="#Author"/>
<rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Description>

<rdf:Description ID="Project Modules">
<rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
<rdfs:domain rdf:resource="#Project"/>
<rdfs:range rdf:resource="#Module"/>
</rdf:Description>

<rdf:Description ID="Author Modules">
<rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
<rdfs:domain rdf:resource="#Author"/>
<rdfs:range rdf:resource="#Module"/>
</rdf:Description>

</rdf:RDF>

```

The actual document describing the resources “A”, “B” and “C” according to that schema may look as follows:

```

<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:software="http://coronet.iicm.edu/rdf/software.rdf#">

<--
  Instances
-->

<software:Module rdf:about="http://coronet.iicm.edu/C" ID="C">
  <software:name>C</software:name>
  <software:language>Java</software:language>
</software:Module>

<software:Project rdf:about="http://coronet.iicm.edu/B">
  <software:name>B</software:name>
  <software:Project Modules>C</software:Project Modules>
</software:Project>

```

```

<software:Author rdf:about="http://coronet.iicm.edu/A">
  <software:name>A</software:name>
  <software:Author Modules>C</software:Author Modules>
</software:Author>

</rdf:RDF>

```

5.2.4 Knowledge Cards on the Semantic Web

Knowledge Cards allows for initial access to hypermedia resources contained in possibly very large data sets. They build up a semantic network of concepts and relationships between these concepts. Hypermedia resources from the underlying data set are assigned to such concepts. Possibility to infer hypermedia resources for each particular concept by means of a set of simple inference rules is one of the main properties of Knowledge Cards. An inference engine built on the top of such Knowledge Cards provides the functionality required for an instant access to best-match hypermedia resources.

Similarities between the Semantic Web and Knowledge Cards are as follows:

- Both apply ontology specification to define terms and relationships between those terms
- Both assign hypermedia resources to the previously defined terms
- Both allow specification of simple infer rules to infer new facts from already existing ones.

However, there exist a number of differences between these two approaches:

- Knowledge Cards are much less expressive than RDF. Hypermedia resources are assigned to Knowledge Cards (i.e., concepts) but they (i.e., resources) are not interrelated by means of semantic relationships. This is true only for concepts. In RDF it is possible to explicitly interrelate resources, which are already assigned to concepts, by means of semantic relationships defined for these concepts.
- Knowledge Cards are designed to solve the problem inherent in large hypermedia applications (that of initial access to best-match hypermedia resources), RDF in turn is much more flexible and expressive.

The conclusion here is that Knowledge Cards are best used to solve the problem of initial access to hypermedia resources inherent to large hypermedia applications. However, that tool might be extended to provide its full interoperability with the Semantic Web, and possible other applications based on similar concepts. Similar to a Knowledge Domain Schema, a particular configuration of Knowledge Cards might be mapped to RDF model to provide RDF based ontology for the Semantic Web. Such ontology would define concepts and their interrelationships existent in that particular Knowledge Cards configuration. A particular set of inference rules should accompany the RDF ontology in order to enable potential software agents to implement an inference engine for hypermedia resources.

Again, a RDF schema defining mapping mechanism between the mapped Knowledge Cards ontology and other ontology based conceptual specifications on the Semantic Web might provide a required level of interoperability between Knowledge Card and these other tools.

Finally, a particular Knowledge Cards assignment might be dumped into an RDF document. Such document would contain actual assignments of resources to different concepts and would represent a base of facts, which could be used by software agents to infer new facts, i.e., to infer assignments of resources for each particular concept within the defined ontology.

5.2.5 Example of mapping Knowledge Cards onto RDF

Let us consider the following example: Knowledge Cards “Hypermedia” and “Databases” are related by means of “is-a-part-of” relationship with Knowledge Card “Computer Science”. Further, Knowledge Cards “Network data model” and “relational data model” are interrelated by means of the same “is-a-part-of” relationship with the Knowledge Card “Databases”.

The following hypermedia resources are assigned to these Knowledge Cards:

- Hypermedia composites “A” and “B” are assigned to “Databases “ Knowledge Card
- Hypermedia composite “C” and an individual HTML document “D” are assigned to the Knowledge Card “Hypermedia”
- PDF document “E” is assigned to the Knowledge Card “Network data model”
- Windows Word document “F” is assigned to the Knowledge Card “Relational data model”.

Finally, a simple inference rule is defined as follows: if a Knowledge Card “A” is related with a knowledge Card “B” by means of semantic relationship “is-a-part-of” then all resources (assigned and inferred) of the Knowledge Card “A” are automatically inferred for the Knowledge Card “B”. This is true for all Knowledge Cards.

By applying that simple infer rules we end up with the following assignment of resources:

- Knowledge Card “Network data model” is accompanied with the resource “E”
- Knowledge Card “Relational data model” is accompanied with the resource “F”
- Knowledge Card “Hypermedia” is accompanied with the resources “C” and “D”
- Knowledge Card “Databases” is accompanied with the resources “A”, “B”, “E” and “F”
- Knowledge Card “Computer Science” is accompanied with the resources “A”, “B”, “C”, “D”, “E” and “F”.

To map the Knowledge Card assignment onto RDF model we could create the following two documents:

- Document defining the RDF schema (including the definition of ontology as well as the definition of the simple inference rule)
- RDF document describing the resources according to the RDF schema.

Here is an example of a RDF schema introducing the terms from the above example:

```
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<!--Definition of Knowledge Card-->
<rdf:Description ID="Knowledge Card">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<!--Relationships-->
<rdf:Description ID="assigned resource">
  <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#Knowledge Card"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<rdf:Description ID="is a part of">
  <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#Knowledge Card"/>
  <rdfs:range rdf:resource="#Knowledge Card"/>
</rdf:Description>
```

Note that the above example does not include the definition of the inference rules nor actual semantic network of Knowledge Cards. The semantic network will be defined in a RDF document accompanied with a particular assignment of hypermedia resources. The definition of inference rules is omitted because of its complexity. However, a real-life mapping mechanism should incorporate such inference rules in RDF schema definitions. The actual document describing the resources “A”, “B”, “C”, “D”, “E” and “F”, as well as the actual semantic network according to the above-defined schema may look as follows:

```
<rdf:RDF xml:lang="en"
```

```

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cs="http://coronet.iicm.edu/rdf/cs.rdf#">

<!--Semantic Network-->
<cs:Knowledge Card ID="Computer Science">
</cs:Knowledge Card>

<cs:Knowledge Card ID="Databases">
  <cs:is a part of>Computer Science</cs:is a part of>
  <cs:assigned resource rdf:about="http://coronet.iicm.edu/A"/>
<cs:assigned resource rdf:about="http://coronet.iicm.edu/B"/>
</cs:Knowledge Card>

<cs:Knowledge Card ID="Hypermedia">
  <cs:is a part of>Computer Science</cs:is a part of>
  <cs:assigned resource rdf:about="http://coronet.iicm.edu/C"/>
  <cs:assigned resource rdf:about="http://coronet.iicm.edu/D"/>
</cs:Knowledge Card>

<cs:Knowledge Card ID="Relational data model">
  <cs:is a part of>Databases</cs:is a part of>
  <cs:assigned resource rdf:about="http://coronet.iicm.edu/F"/>
</cs:Knowledge Card>

<cs:Knowledge Card ID="Network data model">
  <cs:is a part of>Databases</cs:is a part of>
  <cs:assigned resource rdf:about="http://coronet.iicm.edu/E"/>
</cs:Knowledge Card>

</rdf:RDF>

```

5.3 Topic Maps

Topic maps are an ISO standard for describing knowledge structures and associating them with information resources [Pepper, 2000]. There are ongoing initiatives to provide a specification of an XML grammar for interchanging Web-based topic maps. A version 1.0 of XML Topic Maps (XTM) [XTM, 2001] has been already developed.

Topic maps started as a way of representing the knowledge structures inherent in traditional back of book indexes, in order to solve the information management problems involved in creating, maintaining and processing indexes for complex documentation. As the model evolved, their scope was broadened to encompass other kinds of navigational aid, such as glossaries, thesauri and cross-references [Pepper, 2000].

Obviously, topic maps might be used to provide general overviews of the knowledge structures in a hypermedia database, i.e., in a hyperweb. By applying the XTM interchange syntax topic maps might become compatible with the current Web technology.

Let us investigate the basic concepts of topic maps more closely. Topic maps are based on three main concepts (the TAO of Topic Maps [Pepper, 2000]):

- Topics
- Occurrences
- Associations

Topics represent concepts from a particular domain. The topics may be typed, i.e., there may be a whole class hierarchy of different topics, having one topic as a super-class and another one as a subclass topic. Topic has names and a number of other different properties.

Occurrences assign information resources to different topics, i.e., we may say that a particular information resource (say a Web page) is an occurrence of a particular topic. One and the same information resource may be an occurrence of different topics.

Associations relate topics on a meta-level, and indirectly relate all their occurrences. They are used to describe relationships between different topics.

A typical XML topic map (XTM) document includes the definition of topics, associations and occurrences in one XML file.

Now let us look closely on relations between topic maps and semantic data modeling approaches that were presented in this thesis.

5.3.1 Topic maps and Knowledge Domains

Basically, Knowledge Domains and topic maps apply similar concepts in order to provide a representation of the knowledge structures in a hyperweb.

Knowledge Domain Schema defines a number of categories, their properties and their relationships to other categories. Similar to that, topic maps introduce a number of topics and their associations to other topics within that topic map.

In both cases we might talk about a definition of domain ontology that is applied for providing information resources from the underlying database with useful additional information (which might be browsed or queried). In the case of topic maps information resources are defined as occurrences of certain topics, whilst in the case of Knowledge Domains they are defined as instances of particular categories. The last sentence leads to a number of important conclusions, which actually identify important differences between these two approaches.

Thus, the following important modeling elements differentiate these two approaches:

- Knowledge Domains support inheritance, i.e., instances of categories inherit all properties defined for these categories within a Knowledge Domain Schema. For instance, an instance of a category is actually related to another instance of another category. In contrast, topic maps provide facilities to relate just topics on a meta-level, but not its occurrences on their level, i.e., there is no real inheritance in topic maps approach.
- Domain ontology defined by means of a Knowledge Domain Schema is completely separated from an actual assignment of resources to particular categories, i.e., schema is separated from a Knowledge Domain itself. This utilizes the most important aspect of data modeling: separation of structure and content of a database. On the other hand, a topic map defines both: domain ontology and occurrences of topics in information resources within a single topic map. Recently, there has been a lot of development put into defining so-called “topic map templates” mechanism, which would allow for defining domain ontology within one topic map. Such “topic map template” could be used to provide the definition of domain ontology for a number of actual assignments of information resources within different topic maps. However, this mechanism is already ingredient part of Knowledge Domain concept.

5.3.2 Topic maps and Knowledge Cards

Topic maps are a very powerful and general approach to providing an access point to information resources distributed on the Web, in different databases, file systems, etc. As such, this concept may not only be used to semantically model a hypermedia database, but to model any information system. Because of this expressiveness and power the topic maps are sometimes called the “GPS of the information universe”.

Similarly, Knowledge Cards are intended to solve problems of initial access to best-match hypermedia resources from a hypermedia database. Knowledge Cards model the knowledge structures within a

hypermedia databases by similar principles as topic maps do. Thus, a Knowledge Card (concept description) is related to another Knowledge Card by means of semantic relationships. Resources are assigned to Knowledge Cards in the same way as information resources “occur” as particular topics in a topic map.

Differences between these two modeling approaches are more of behavioral nature. Namely, Knowledge Cards are equipped with an inference engine that exploits such important property of the underlying data structures (semantic networks) such as possibility to infer new facts from already existing facts. Thus, accessing a Knowledge Card means accessing all resources assigned to that particular Knowledge Card but also accessing all resources inferred for that particular Knowledge Card. Although, such mechanism might be easily defined by means of a topic map [ref], this mechanism might be considered as an additional tool that is by no means ingredient part of a topic map itself.

Another important difference between these two approaches considers authoring of topic maps and Knowledge Cards. For instance, authors might define their own personal Knowledge Card and relate that Knowledge Card with Knowledge Cards describing concepts where they want to contribute. Now, authors might put their resources in their own personal Knowledge Card and let the inference engine infer these resources to related Knowledge Cards. In that way authoring is highly simplified in contrast to topic maps where authors should know the whole topic map in order to assign their resources on the right place.

6 Conclusion

First of all, let us look at a short summary of this thesis. In this thesis I considered the World Wide Web to be the largest library of human knowledge that mankind has ever created. Any kind of information such as daily news, scientific research reports, sport results, movie reviews, educational materials, literally anything, can be found on the Web. Unfortunately, many Web users experience, even at a daily basis, serious problems with the Web. Links are broken, navigational tools are inconvenient, finding relevant piece of information is a tedious job, the presented knowledge is unstructured, etc.

I claimed that such problems exist because the Web, i.e., the largest distributed knowledge library does not support knowledge transfer processes, such as knowledge structuring, knowledge mining or knowledge profiling.

Speaking more technically, the Web is a large distributed hypermedia system. All hypermedia systems are just special kind of database systems, thus they commit to a certain data model.

Thus, in order to provide a structured technical analysis of problems of the Web I looked on it from data modeling point of view. I applied the well-known three level data modeling architecture, thus identifying the physical, logical and semantic data model of the Web.

I did not analyze in details the physical data model of the Web because such an analysis was out of scope of this thesis. Rather I investigated the logical and semantic data model of the Web.

Thus, I provided the detailed overview of the logical data model of the Web, as well as the current trends in logical data modeling for the Web. This overview listed advantages and disadvantages of such modeling approaches. I showed that almost entire problems of the Web are caused by inappropriate logical data modeling paradigms as found in the Web nowadays. Logical data modeling solutions to certain (but limited) number of problems of the Web were presented in this overview as well. Further, I showed that semantic data modeling is not supported on the Web at all

The purpose of the overview of the logical data modeling on the Web was to show that it is not possible to support knowledge transfer processes on the Web (and thus to solve a large number of its problems) solely by means of logical data modeling paradigms. I proved that such processes might be supported only if we apply semantic data modeling to the Web.

Thus, I presented semantic data modeling concepts that I developed in past few years, which support knowledge transfer processes such as knowledge structuring on the Web, knowledge profiling on the Web and knowledge mining on the Web. These concepts are HC-Data model, Knowledge Domains and Knowledge Cards, respectively. Each of these concepts was discussed in details analyzing its advantages, disadvantages and possible improvements. Moreover, implementation of such approaches in a Web environment was presented as well.

Finally, I listed similar data modeling approaches such as Resource-description-framework and topic maps and compared them with the presented approaches.

To conclude this work I would like to make few suggestions for further development.

First of all, visualization of Knowledge Domains must be enhanced with modern visualization techniques for semantic networks. Such graphic user interface component providing an access point to a particular Knowledge Domain would allow users to see at a glance the knowledge profile of that Knowledge Domain. This in turn would facilitate an improved feeling of users' orientation within that Knowledge Domain. Currently, such graphical overview exists only for depicting a particular Knowledge Domain Schema. Obviously, that must be extended to provide a graphical overview of a complete Knowledge Domain including all instances of categories defined by that particular Knowledge Domain Schema.

The expressive power of Knowledge Domain Schema language must be improved allowing for defining not only simple 1:n relations but more general m:n relations. That would provide the possibility to model real-life situations more closely. Further, the language of Knowledge Domain Schema must support defining of relations of any arity, not only of binary relations. Again, such facility provides for defining of models, i.e., knowledge profiles that are much closer to real-world situations.

Further, building semi-automatic or even automatic software agents that would assign hypermedia resources to predefined categories could be very interesting research topic. Analyzing not only meta-data attached to hypermedia resources, but also textual (or even any other kind) information contained in it should be considered. In the case of semi-automatic procedures knowledge engineers should have their impact on creating of a particular assignment. For instance, system might collect and analyze data and prompt knowledge engineers to select one or more categories to which a particular resource should be assigned.

Finally, I believe that the concepts of Knowledge Domains and Knowledge Cards should be generalized in order to provide a unified concept. That means that the underlying objects from both models, i.e., semantic categories, relationships and instances form Knowledge Domains, and Knowledge Cards, relationships and resources from Knowledge Card configurations should be abstracted to more general object types that might be manipulated using a common interface. For instance, such approach allows for defining and applying a simple set of inference rules for a particular Knowledge Domain or a particular Knowledge Cards configuration (or whatever a unified name for these concepts might be) transparently of a particular model. Thus, one and the same inference engine might be used for both Knowledge Domains and Knowledge Cards. This greatly facilitates the approach of the Semantic Web and Resource-description-framework. Such approach supports the interoperability of different software programs on the Web at the level of different conceptualizations of the underlying Web-based knowledge structures.

Bibliography

- [Al-Khatib et al., 1999] W. Al-Khatib, Y. Francis Day, A. Ghafoor, P. B. Berra: *Semantic modeling and knowledge representation in multimedia databases*, IEEE Transactions on Knowledge and Data Engineering, 11(1), 64-80, 1999.
- [Andrews et al., 1995] K. Andrews, A. Nedoumov, N. Scherbakov: *Embedding Courseware Into Internet: Problems and Solutions*, Proceedings of ED- MEDIA'95, Graz, Austria, 69-74, 1995
- [Andrews, 1996] K. Andrews: *Browsing, Building and Beholding Cyberspace: New Approaches to the Navigation, Construction and Visualization of Hypermedia on the Internet*, PhD. Thesis, Graz University of Technology, Austria, 1996.
- [Arents and Bogaerts, 1996] H.C. Arents, W. F. L. Bogaerts: *Concept-Based Indexing and Retrieval of Hypermedia Information*, Encyclopedia of Library and Information Sciences (supplement volumes), Vol. 58, 1-29, Academic Press, New York, 1996.
- [ASP, 2001] *Active Server Pages (General)*,
<http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000440>
- [Barwise and Etchemenedy, 1990] J. Barwise, J. Etchemenedy: *Valid Inference and Visual Representation*, Zimmerman and Cunningham (Ed.), Visualization in Mathematics, American Mathematics Association, 1990.
- [Berners-Lee et al., 1992] T. Berners-Lee, R. Cailliau, J. F. Groff, B. Pollermann: *World-Wide Web: The Information Universe*, Electronic Networking, Research, Applications and Policy 1(2), 74-82, 1992.
- [Berners-Lee et al., 1992a] T. Berners-Lee, R. Cailliau, J. F. Groff: *The World-Wide Web*, Computer Networks and ISDN Systems 25(4-5), 454-459, 1992.
- [Berners-Lee, et al. 1994] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, A. Secret: *The World-Wide Web*, Communications of the ACM, 37(8), 76-82, 1994.
- [Berners-Lee and Fischetti, 1999] T. Berners-Lee, M. Fischetti: *Weaving the Web: the Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*, Harper, 1999.
- [Berners-Lee et al., 2001] T. Berners-Lee, J. Hendler, O. Lassila: *The Semantic Web*, Scientific American, 2001.
- [Bernstein 1991] M. Bernstein: *The navigation problem reconsidered*, Hypertext/Hypermedia Handbook, E. Berk, J. Devlin (Ed.), Mc-Graw-Hill, New York, 1991.
- [Borgida, 1987] A. Borgida: *Conceptual Modeling of Information Systems*, On Knowledge Base Management Systems, Springer-Verlag, M. L. Brodie, J. Mylopoulos (Ed.), 1987.

- [Borgida, 1991] A. Borgida: *Knowledge Representation, Semantic Modeling: Similarities and Differences*, E-R Approach '91, Elsevier Publishing, 1991.
- [Brodie et al., 1984] M. Brodie, J. Mylopoulos, J. Schmidt (Ed.): *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer Verlag, New York, NY, 1984.
- [Brusilowski and Schwarz, 1997] P. Brusilowski, E. Schwarz: *Concept-based Navigation in Educational Hypermedia and its Implementation on WWW*, T. Mueledner, T. C. Reeves (Ed.): *Educational Multimedia/Hypermedia and Telecommunications*, 112-117. University of Calgary, AACE, 1997.
- [Brusilowski et al., 1998] P. Brusilowski, J. Eklund, E. Schwarz: *Web-based Education for all: a Tool for the Development of Adaptive Courseware*, *Computer Networks and ISDN Systems*, 30, 1-7, 1998.
- [Bush, 1945] V. Bush: *As We May Think*, *The Atlantic Monthly*, 176(1), 101-108, 1945.
- [Calvanese et al., 1998] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati. *Knowledge representation approach to information integration*, *Proceedings of AAAI Workshop on AI and Information Integration*, 58-65, AAAI Press/The MIT Press, 1998.
- [Calvanese et al., 1998a] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati: *Information Integration: Conceptual Modeling and Reasoning Support*. *Proceedings of CoopIS 1998*, 10, 280-291, 1998.
- [Canter et al., 1985] D. Canter, R. Rivers, G. Storrs: *Characterizing User Navigation Through Complex Data Structures*, *Behaviour and Information Technology*, 4(2), 93-102, 1985.
- [Carmody et al., 1969] S. Carmody, W. Gross, T. H. Nelson, D. Rice, A. van Dam: *A Hypertext Editing System for the 360*, Faiman, Nievergelt (Ed.): *Pertinent Concepts in Computer Graphics*, University of Illinois Press, 291-330, 1969.
- [Castells and Szekely, 1999] P. Castells, P. Szekely: *Presentation Models by Example*, D. J. Duke, A. Puerta (Ed.): *Design, Specification and Verification of Interactive Systems '99*, Springer Verlag, 1999.
- [Chang et al., 1986] S. K. Chang, T. Ichikawa, P. A. Ligomenides: *Visual Languages*, New York, Plenum Press, 1986.
- [Codd, 1970] E. F. Codd: *A Relational Model of Data for Large Shared Data Banks*, *Communications of the ACM* 13(6), 377-387, 1970.
- [CoffeeCup, 2001] *CoffeeCup HTML Editor Homepage*, <http://www.coffeecup.com/editor/>
- [Comai et al., 1998] S. Coma, E. Damiani, R. Posenato, L. Tanca: *A Schema-based Approach to Modeling and Querying WWW Data*, *Proceedings of FQAS '98*, Roskilde, 1998.
- [Composer, 2001] *Products & Services*, <http://home.netscape.com/products/index.html>
- [Conklin, 1987] J. Conklin: *Hypertext: An Introduction and Survey*, *IEEE Computer*, 20(9), 17-41, 1987.
- [Corby et al., 2000] O. Corby, R. Dieng, C. Hebert: *A Conceptual Graph Model for W3C Resource Description Framework*, *ICCS 2000 - International Conference on Conceptual Structures*, Darmstadt, Germany, Springer, 2000.
- [CSS, 2001] *Cascading Style Sheets*, <http://www.w3.org/Style/CSS/>
- [DCMI, 2001] *Dublin Core Metadata Initiative (DCMI)*, <http://dublincore.org/>

- [Decker et al., 1999] S. Decker, M. Erdmann, D. Fensel, R. Studer: *Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information*, R. Meersman et al. (Ed.), Database Semantics: Semantic Issues in Multimedia Systems, 51-369. Kluwer Academic Publisher, 1999.
- [De Young, 1990] L. De Young: *Linking Considered Harmful*, Hypertext: Concepts, Systems and Applications, Proceedings of European Conference on Hypertext, Versailles France, 238-249, 1990.
- [Dillon and Tan, 1993] T. Dillon and P.L. Tan: *Object-Oriented Conceptual Modeling*, PrenticeHall, 1993.
- [Dijkstra, 1968] E.W. Dijkstra: *Go to statement considered harmful*, Communications of the ACM, 11(3), 147-148, 1968.
- [Dietinger and Maurer, 1997] T. Dietinger, H. Maurer: *How Modern WWW Systems Support Teaching and Learning*, Proceedings of International Conference on Computers in Education 1997, Z. Halim, T. Ottmann, Z. Razak (Ed.), Kuching, Sarawak Malaysia, 37 – 51, 1997.
- [Dietinger and Maurer, 1998] T. Dietinger, H. Maurer: *GENTLE - (General Networked Training and Learning Environment)*, Proceedings of ED-MEDIA'98, AACE, Charlottesville, VA, 1998.
- [Dreamweaver, 2001] *Macromedia - Dreamweaver*, <http://www.macromedia.com/software/dreamweaver/>
- [Duval et al., 1995] E. Duval, H. Olivie, P. Hanlon, D. Jameson: *HOME: An environment for hypermedia objects*, Journal of Universal Computer Science, 1, 154-173, 1995.
- [Edwards and Hardmann, 1989] D. M. Edwards, L. Hardmann: *Lost in Hyperspace: Cognitive Mapping and Navigation in a Hypertext Environment*, R. McAleese (ed.), Hypertext: Theory and Practice, 105-125, Blackwell Scientific Publications, Ablex, 1989.
- [Engelbart, 1963] D. C. Engelbart: *A Conceptual Framework for the Augmentation of Man's Intellect*, Vistas in Information Handling, P. Howerton, Washington DC, Spartan Books, 1, 1-29, 1963.
- [Engelbart and English, 1968] D. C. Engelbart, W. K. English: *A Research Center for Augmenting Human Intellect*, Proceedings of the Fall Joint Computer Conference (IFJC), 33, Arlington, 395-410, 1968.
- [Explorer, 2001] *Welcome to the Internet Explorer Home Page!*, <http://www.microsoft.com/windows/ie/default.htm>
- [Fensel et al., 1998] D. Fensel, S. Decker, M. Erdmann, R. Studer: *Ontobroker: Or how to make the www intelligent*, Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management, KAW'98, Banff, Canada, 1998.
- [Fensel et al., 1998a] C.Fensel, S. Decker, M. Erdmann, R. Studer: *Ontobroker: The Very High Idea*, Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibel Island, Florida, USA, 131-135, 1998.
- [Fernandez et al., 1997] M. Fernandez, D. Florescu, J. Kang, A. Levy, D. Suci: *Strudel: A Web-site management system*, Proceedings of ACM SIGMOD '97 Tucson, USA, 549-552, 1997.
- [Flash, 2001] *Macromedia - Flash*, <http://www.macromedia.com/software/flash/>
- [FrontPage, 2001] *Microsoft Office - FrontPage Home*, <http://www.microsoft.com/frontpage/>
- [Gaines and Show, 1995] B. Gaines, M. Shaw: *Concept maps as hypermedia components*, International Journal of Human-Computer Studies, 1995.

- [Garzotto et al., 1991] F. Garzotto, P. Paolini, D. Schwabe: *HDM – A Model for the Design of Hypertext Application*, Proceedings of the ACM Hypertext '91, St. Antonio, USA, 47-58, 1991.
- [Garzotto et al., 1996] F. Garzotto, L. Mainetti, P. Paolini: *Information Reuse in Hypermedia Applications*, Seventh ACM Conference on Hypertext, 1996.
- [Google, 2001] *Google*, <http://www.google.com>
- [Gruber, 1993] T. Gruber: *What is an Ontology?*, Knowledge Systems Laboratory, Computer Systems Dept., Stanford University, Stanford, USA, 1993.
- [Gruber, 1993a] T. Gruber: *A Translation Approach to Portable Ontology Specifications*, Knowledge Acquisition, 5(2), 199-220, 1993.
- [Halasz, 1988] F. G. Halasz: *Reflections on NoteCards: Seven Issues for the Next generation of Hypermedia systems*, Communications of the ACM, 31(7), 836-852, 1988.
- [Halasz, 1991] F. G. Halasz: *Seven Issues Revisited*, Final Keynote Talk at the 3rd ACM Conference on Hypertext, San Antonio, USA, 1991.
- [Hammond, 1993] N. Hammond: *Learning With Hypertext: Problems, Principles and Prospects*, C. McKnight, A. Dillon, J. Richardson (Ed.), Hypertext, a psychological perspective, Ellis Horwood Ltd, 1993.
- [Hardmann et al., 1994] L. Hardman, D. C. A. Bulterman, G. van Rossum: *The Amsterdam Hypermedia Model - Adding Time and Context to the Dexter Model*, Communications of the ACM, 37, 50-62, 1994.
- [Helic et al., 1999] D. Helic, H. Maurer, N. Scherbakov. *Introducing Hypermedia Composites to WWW*, Journal of NCA 22(1), 19-32, 1999.
- [Helic et al., 1999a] D. Helic, H. Maurer, N. Scherbakov: *Authoring and Maintaining of Educational Applications on the Web*, Proceedings ED-MEDIA'99, AACE, Charlottesville, USA, 1792-1797, 1999.
- [Helic et al., 1999b] D. Helic, S. Maglajlic, N. Scherbakov: *The HC-Data Model: A New Semantic Hypermedia Data Model*, Proceedings of WebNet' 99, AACE, Charlottesville, USA, 493-498, 1999.
- [Helic et al., 2000] D. Helic, H. Maurer, N. Scherbakov: *Web Based Training: What do we expect from the system*, Proceedings of ICCE 2000, Taiwan 1689-1694, 2000.
- [Helic et al., 2001] D. Helic, H. Maurer, N. Scherbakov: *Accessing Best-Match Learning Resources in WBT Environment*, Proceedings of ED-MEDIA, AACE, 2001.
- [Helic et al., 2001a] D. Helic, H. Maurer, N. Scherbakov: *Knowledge Domains: A Global Structuring Mechanism for Learning Resources in WBT Systems*, Proceedings of WebNet, AACE, 2001.
- [Helic et al., 2001b] D. Helic, H. Maurer, J. Lennon, N. Scherbakov: *Aspects of a Modern WBT System*, Proceedings of SSGRR 2001, Rom, Italy, 2001.
- [HM-Card, 2001] *About HM-Card*, http://coronet.iicm.edu/HM_Tools/hmcard15/hmcard.htm
- [Hotho et al., 2001] A. Hotho, A. Maedche, S. Staab, R. Studer: *Seal-II – The Soft Spot between Richly Structured and Unstructured Knowledge*, Proceedings of I-Know '01, Graz, Austria, 2001.
- [HTML, 2001] *HTML Home Page*, <http://www.w3.org/MarkUp/>
- [HTTP, 2001] *HTTP - HyperText Transfer Protocol Overview*, <http://www.w3.org/Protocols/>

- [Hull and King, 1987] R. Hull, R. King: *Semantic Database Modelling: Survey, Applications, and Research Issues*, ACM Computing Surveys 19(3), 201-260, 1987.
- [Hypercard, 1989] *Hypercard*, Hypercard Reference Manual, Apple Computer Inc. Cupertino USA, 1989.
- [HyperWave, 2001] *HyperWave - The Power of Wisdom*, <http://www.hyperwave.com/index.html>
- [IMS, 2001] *IMS Specifications: Meta-Data*, <http://www.imsproject.org/metadata/index.html>
- [Kappe, 1991] F. Kappe: *Aspects of a Modern Multi-Media Information System*, PhD. thesis, Graz University of Technology, Austria, 1991.
- [Kappe, 1993] F. Kappe: *Hyper-G: A Distributed Hypermedia System*, B. Leiner (Ed.), Proceedings of INET '93, San Francisco, USA, DCC-1-DCC-9, Internet Society, 1993.
- [Kappe 1995] F. Kappe: *Hypermedia Systems: Hyper-G now Hyperwave*, H. Maurer (Ed.), Addison-Wesley, 1995.
- [Lassila and Swick, 1999] O. Lassila, R. R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation, 22 February 1999.
- [Lehman, 1992] F. Lehman (Ed.): *Semantic Networks in Artificial Intelligence*, Oxford: Pergamon Press A collection papers containing recent work within the area of semantic networks and includes both applied and theoretical information, 1992.
- [Lennon, 1997] J. Lennon: *Hypermedia Systems and Applications – World Wide Web and Beyond*, Springer Berlin, 1997.
- [Lipman, 1980] A. Lipmann: *Movie-Maps: An Application of the Optical Videodisc to Computer Graphics*, Computer Graphics, 14(3), 32-42, 1980.
- [Luke et al., 1997] S. Luke, L. Spector, D. Rager, J. Hendler: *Ontology-based Web agents*, Proceedings of the First International Conference on Autonomous Agents New York, 1997.
- [Maglajlic et al., 1999] S. Maglajlic, D. Helic, N. Scherbakov: *Generating the Web Sites Using Templates and Text-patterns*, Proceedings of ED-MEDIA '99, Seattle, USA; Association for the Advancement of Computing in Education, Charlottesville, 1999.
- [Maurer et al., 1994] H. Maurer, A. Philpott, N. Scherbakov: *Hypermedia Systems without Links*, Journal of Microcomputer Applications, 17(4), 1994.
- [Maurer et al., 1994a] H. Maurer, N. Scherbakov, K. Andrews, S. Parthasarathy: *Object-oriented Modeling of Hyperstructure: Overcoming the Static Link Deficiency*, Information and Software Technology, 36(6), 315-322, 1994.
- [Maurer, 1996] H. Maurer (Ed.): *HyperWave: The Next Generation Web Solution*, Addison-Wesley, 1996.
- [Maurer et al., 1996] H. Maurer, N. Scherbakov, K. Andrews: *Browsing Hypermedia Composites: An Algebraic Approach*, Proceedings of WebNet '96, San Francisco, USA, 1996.
- [Maurer and Scherbakov, 1996] H. Maurer, N. Scherbakov: *Multimedia Authoring for Presentation and Education: The Official Guide to HM-Card*, Addison-Wesley Bonn, 1996.
- [Maurer et al., 1998] H. Maurer, N. Scherbakov, Z. Halim, Z. Razak: *From Databases to Hypermedia*, Springer Berlin, 1998.

- [McAleese, 1989] R. McAleese: *Navigation and Browsing in Hypertext*, McAleese R. (Ed.), Hypertext: Theory into Practice, 6-44, Blackwell Scientific Publications, Ablex, 1989.
- [Minsky, 1968] M. Minsky: *Semantic Information Processing*, Cambridge, MA: MIT Press, 1968.
- [Minsky, 1975] M. Minsky: *A framework for representing knowledge*, P. Winston (Ed.): The Psychology of Computer Vision. New York: McGraw Hill, 1975.
- [Mosaic, 1993] *NCSA Mosaic Home Page*, <http://archive.ncsa.uiuc.edu/SDG/Software/Mosaic/>
- [Navigator, 2001] *Products & Services*, <http://home.netscape.com/products/index.html>
- [NCSA, 2001] *Welcome to NCSA*, <http://www.ncsa.uiuc.edu/>
- [Nelson, 1965] T. H. Neslon: *A File Structure for the Complex, the Changing, and the Indeterminate*, Proceedings of ACM 20th National Conference, 84-100, 1965.
- [Nielsen, 1990] J. Nielsen: *Hypertext and Hypermedia*. Academic Press Inc. Boston, 1990.
- [Nosek and Roth, 1990] J. T. Nosek, I. Roth: *A comparison of formal knowledge representations as communication tools: predicate logic vs semantic network*, International Journal of Man-Machine Studies, 33, 227-239, 1990.
- [Nyce and Kahn, 1991] J. M. Nyce, P. Kahn (Ed.): *From Memex to Hypertext: Vannevar Bush and the Mind's Machine*, Academic Press Inc. Boston, 1991.
- [Parunak, 1991] H. Van Dyke Parunak: *Don't Link Me In: Set Based Hypermedia for Taxonomic Reasoning*, Hypertext 1991, 233-242, 1991.
- [PDF, 2001] *Adobe PDF*, <http://www.adobe.com/products/acrobat/adobepdf.html>
- [Peckham and Maryanski, 1988] J. Peckham, F. Maryanski: *Semantic database models*, ACM Computing Surveys, 20(3), 153-189, 1988.
- [Pepper, 2000] S. Pepper: *The TAO of Topic Maps*, XML Europe 2000, www.gca.org/papers/xml europe2000/papers/s11-01.html.
- [Perez et al., 1995] T. A. Perez, J. Gutierrez, P. Lopisteguy: *An Adaptive Hypermedia System*, Proceedings of Artificial Intelligence in Education, AACE, 1995.
- [PHP, 2001] *PHP: Hypertext Preprocessor*, <http://www.php.net/>
- [PowerPoint, 2001] *Microsoft PowerPoint Version 2002 Documentation*, <http://www.microsoft.com/office/techinfo/productdoc/2002/en/powerpoint/>
- [Raging, 2001] *Raging*, <http://www.altavista.com/sites/search/text?raging=1>
- [Ramaiah, 1992] C. K. Ramaiah: *An Overview of Hypertext and Hypermedia*, International Information, Communication & Education 11(1) 26-42, 1992.
- [RDF, 2001] *W3C Semantic Web Activity: Resource Description Framework (RDF)*, <http://www.w3.org/RDF/>
- [Rivlin et al., 1994] E. Rivlin, R. A. Botafogo, B. Shneiderman: *Navigating in Hyperspace: Designing a Structure-Based Toolbox*, Communications of the ACM, 37(2): 87-96, 1994.

- [Rishe, 1992] N. Rishe: *Database Design The Semantic Modeling Approach*, McGraw-Hill, New York, 1992.
- [Semantic Web, 2001] *W3C Semantic Web Activity*, <http://www.w3.org/2001/sw/>
- [Simons, 1987] P. Simons: *Parts: A study in Ontology*, Clarendon Press, Oxford, 1987.
- [Sowa, 1984] J. F. Sowa: *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, 1984.
- [Sowa, 1991] J. F. Sowa (Ed.): *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, San Mateo, USA, Morgan-Kaufman, 1991.
- [SSL, 2001] *SSL 3.0 Specification*, <http://www.netscape.com/eng/ssl3/>
- [Staab and Maedche, 2001] S. Staab, A. Maedche. *Knowledge portals - ontologies at work*. AI Magazine, 21(2), 2001.
- [Streitz et al., 1992] N. Streitz, J. Haake, J. Hanneman, A. Lemke, W. Schuler, H. Schutt, M. Thuring: *SEPIA: A Cooperative Authroing Environment*, Proceedings of the ACM ECHT '92, Milan, Italy, 11-22, 1992.
- [Tomek et al., 1991] I. Tomek, S. Khan, T. Muldner, M. Nassar, G. Novak, P. Proszynski: *Hypermedia - Introduction and Survey*, Journal of Microcomputer Applications, 14(2), 63-103, 1991.
- [Tsichritzis and Klug, 1978] D. Tsichritzis, A. C. Klug: *The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Dabatase Management Systems*, IS 3(3), 173-191, 1978.
- [Tsichritzis and Lochovsky, 1982] D. Tsichritzis, F. Lochovsky: *Data Models*, Prentice-Hall, 1982.
- [URL, 2001] *Web Naming and Addressing Overview*, <http://www.w3.org/Addressing/>
- [WBT-Master, 2001] *WBT-Master*, <http://coronet.iicm.edu/>
- [WinWord, 2001] *Microsoft Word Version 2002 Documentation*, <http://www.microsoft.com/office/techinfo/productdoc/2002/en/word/>
- [XML, 2001] *Extensible Markup Language*, <http://www.w3.org/XML/>
- [XTM, 2001] *XML Topic Maps (XTM) Home Page*, <http://www.topicmaps.org/xtm/index.html>
- [Yankelovich et al., 1988] N. Yankelovich, B. J. Haan, N. K. Meyrowitz, S. M. Drucker: *Intermedia: The Concept and the Construction of a Seamless Information Environment*, IEEE Computer, 21(1), 81-96, 1988.
- [Zellweger, 1989] P. T. Zellweger: *Scripted Documents: A Hypermedia Path Mechanism*, Proceedings of the ACM Hypertext '89, Pittsburgh, USA, 1-14, 1989.