Systematic Hypermedia Design

V. Balasubramanian
E-Papyrus, Inc.,
63E Reading Road
Edison, NJ 08817 and
Graduate School of Management
Rutgers University
Newark, NJ 07102.
Tel/Fax: 1-908-548-7868
Email: *bala@pegasus.rutgers.edu*
URL: *http://eies.njit.edu/~333/bala.html*

Michael Bieber
Institute of Integrated Systems Research
Department of Computer and Information Science
New Jersey Institute of Technology
Newark, NJ 07102.
Tel: 1-201-596-2681
Fax: 1-201-596-5777
Email: *bieber@cis.njit.edu*
URL: *http://hertz.njit.edu/~bieber/bieber.html*

Tomás Isakowitz
Information Systems Department
Leonard N. Stern School of Business
New York University
New York, NY 10012.
Tel: 1-212-998-0833
Fax: 1-212-995-4228
Email: *tomas@stern.nyu.edu*
URL: *http://www.stern.nyu.edu/~tisakowi*

5/10/96

Systematic Hypermedia Design

ABSTRACT

Hypermedia structuring and navigation requires design methodologies different from those developed for standard information systems. This article details our successful application of Relationship Management Methodology (RMM), a hypermedia systems analysis and design methodology, to ACM SIGLINK's LINKBase. LINKBase is a World-Wide Web (WWW) application, which dynamically generates WWW pages from a relational database containing information about hypertext-related events such as conferences, publications, authors, and sponsoring organizations. We describe our experience applying RMM in this case study, summarize design lessons we learned in the process, present extensions to RMM, and ground our work in the hypermedia design literature. Our experiences should encourage hypermedia and WWW developers to utilize systematic design techniques.

KEYWORDS

Hypertext, Hypermedia, Relational Database Display, Hypermedia design methodology, Design guidelines, Entity-Relationship Diagrams, Navigation, Indexes, Guided Tours, World-Wide Web database gateways, LINKBase, Information Access.

KSRL categories: AI0102, CB0601.05, CB0902.01, FB04, FC10, HA0903

ACKNOWLEDGMENTS

We would like to thank Bang-Min Ma, Yu Cheng, Sajeev Joseph at the New Jersey Institute of Technology, Joonhee Yoo at Rutgers University and Mark Ginsburg at New York University. We gratefully acknowledge funding for this project by NJIT under Grant #990967, by the National Center for Transportation and Industrial Productivity under Grant #990905, grants from the Sloan Foundation and the AT&T Foundation, and the NASA JOVE faculty fellowship program.

Biographies of Authors

V. Balasubramanian ("Bala") has nine years experience in developing information systems. Currently, he heads his own consulting firm, E-Papyrus, Inc., specializing in hypermedia design, Web-based document delivery, document management, groupware and user interface design. Most recently, Bala was instrumental in introducing Web-based information services at Hoffmann-La Roche, a pharmaceutical company. He has consulted for IBM, Bell Atlantic, and AT&T before. He is also a doctoral candidate at the Graduate School of Management, Rutgers University.

Michael Bieber is Assistant Professor of Information Systems at the New Jersey Institute of Technology, where he directs the Hypermedia Information Systems Lab. He is a Faculty Fellow in the Automation Technology Section at NASA's Goddard Space Flight Center at NASA. He is active in the hypertext research community. With Tomás Isakowitz he co-organizes the Hypermedia in Information Systems and Organizations minitrack at the annual HICSS conferences and has co-edited several special journal issues on hypermedia. He serves as ACM SIGLINK's treasurer. He holds a Ph. D. in Decision Sciences from the University of Pennsylvania.

Tomás Isakowitz is an Assistant Professor of Information Systems at New York University Stern School of Business. His research interests are hypermedia technology and its applications, decision support and temporal databases. Professor Isakowitz received his B.Sc. in Mathematics at the Hebrew University of Jerusalem, his M. Sc. in Mathematics at the University of California at Santa Barbara, his M. Eng. and Ph.D. in Computer Science at the University of Pennsylvania. He has taught at New York University's Stern School of Business, the International University of Japan, and the University of Pennsylvania, and was a visiting scholar at Stanford Research Institute. Dr. Isakowitz' software engineering research interests focus on exploring new paradigms and methodologies to facilitate software development, encourage software reuse and encourage solid system construction. He is actively involved in hypertext research, and has written extensively about the design and development of hypertext/hypermedia applications. His publications have appeared in the *Journal of Management Information Systems, Communications of the ACM, Decision Support Systems, ACM Transactions on Database Systems, ACM Transactions on Office Information Systems, Decision Support Systems* and elsewhere. He currently serves on the editorial boards of the *Journal of Management Information Systems*, the *Journal of Electronic Commerce,* and as a special issue editor for the *Journal of Organizational Computing* and the *Communications of the ACM*. Dr. Isakowitz has worked as a consultant for several international, and is actively involved in the academic and practical issues involving the design of hypermedia applications.

Systematic Hypermedia Design

1. INTRODUCTION

Organizations increasingly are seeking to exploit hypermedia[1] to augment users' access to information. For example, we find hypertext-based information kiosks in museums, airports and other public places. Hypertext techniques help software developers find program segments for reuse (Isakowitz and Kauffman 1997). Hypermedia structuring gives executives and analysts ready access to details and explanations within decision support systems (Kimbrough, Pritchett, Bieber and Bhargava 1990; Mao, Benbasat and Dhaliwal 1996; Minch 1990; Minch and Green 1996). On the World Wide Web (WWW), organizations are exploring the potential of hypertext access to present themselves to the public, and to sell their products and services. Yet, while the field of hypermedia itself is maturing, methodologies for hypermedia design only now are beginning to emerge. To date, no published work has employed a systematic hypermedia design methodology for WWW development.

This article details our successful application of a hypermedia systems analysis and design methodology to a WWW application. We present a case study using the Relationship Management Methodology (RMM) by Isakowitz, Stohr and P. Balasubramanian (1995), propose extensions to RMM, summarize the "RMM design" lessons we learned in this process, and ground our experience in the hypermedia design literature. Our experiences should help pave the way for other developers to take advantage of hypermedia design and development techniques, both on the WWW and in more traditional applications.

What makes hypermedia so useful? We view hypermedia as the science of relationships and hypermedia techniques as supporting *relationship management*. Hypermedia concerns structuring, presenting and giving users direct access to the content and interconnections within an information domain. A hypermedia vantage point encourages designers to consider a system in terms of the relationships among its objects and processes, focusing on what users might want to access and how users should access them.

Handcrafting hypermedia interrelationships is tedious at best, and impractical in cases of voluminous or frequently changing information. Furthermore, authoring a hypermedia interface and hypermedia links in an *ad hoc* format leads to inconsistent design (Garzotto, Mainetti and Paolini 1995), and the possibility of errors and omissions (Botafogo, Rivlin and Shneiderman 1992; Brown, 1990). Therefore, a systematic design approach with automatic page generation constitutes the only reasonable option in many cases.

(Isakowitz et al., 1995) focused on the first three of RMM's seven stages and provided initial developer guidelines for these. The current article reviews RMM's seven stages, extends some aspects of the first three stages, and describes the remaining four stages in detail. We also present the first full case study of a step-by-step application of this hypermedia design methodology. While many WWW applications also retrieve display contents from databases (see, for example, CommerceNet [URL1] and Johns Hopkins University's public domain bioinformatics databases [URL2]), ours is the first published description of a WWW application designed with a systematic hypermedia design methodology. Our emphasis in the previous sentence is twofold: on *published* and on *systematic hypermedia design*. We assume these other applications resulted from some degree of design. Yet without any published record we do not know how much. And, more important, others can not learn from them.

We begin in §2 by reviewing RMM and our extensions to it. §3 discusses related hypermedia design research. In §4 we introduce our application, LINKBase. §5 presents our case study applying RMM to LINKBase. §6 takes a larger view of RMM and discusses issues such as hypermedia requirements analysis and retrofitting legacy applications. We conclude in §7 with some final motivation.

[1] While the term hypermedia nominally encompasses multiple media, we make no distinction between hypertext and hypermedia in this article.

2. The Relationship Management Methodology (RMM)

The *Relationship Management Methodology (RMM)* addresses the design and construction of hypermedia applications. We begin this section by briefly presenting RMM and its data model, RMDM. A more detailed discussion can be found in (Isakowitz et al., 1995).

2.1 Methodological Steps

RMM consists of the following seven steps, some of which can be conducted in parallel: (1) *Entity-Relationship design*: models the information domain and its relationships, (2) *Slice design*: how information units are sub-divided for display, (3) *Navigational design:* how users will access information, (4) *User-Interface Design:* how information will be presented, (5) *Protocol Conversion Design:* how abstract constructs are to be transformed into physical-level constructs, e.g., what kind of WWW page corresponds to an index, (6) *Run-time behavior:* how to populate the application with data, and (7) *Construction and testing.*

Although first presented as a linear methodology RMM was conceived to be flexible by supporting rapid feedback loops as prescribed in (Nanard and Nanard, 1995). Research in this direction has been embodied in software design tools presented in (Diaz and Isakowitz, 1995).

2.2 The RMDM Data Model

The Relationship Management Data Model (RMDM) is the cornerstone of the RMM methodology. Figure 1 presents its elements. RMDM includes elements for representing information domain concepts (such as entities and relationships), and navigation mechanisms (such as links). An application's design is described via an RMDM diagram (see Figure 9 on page 15). The RMDM model is based on the Entity-Relationship model (Elmasri and Navathe, 1990), and on HDM (Garzotto, Paolini and Schwabe 1993) and HDM2 (Garzotto, Mainetti and Paolini 1996).

Because entities may have a large number of attributes of a different nature (e.g., salary information, biographical data, photograph), it may be impractical or undesirable to present all the attributes of an entity instance in one screen. Thus, RMM groups attributes into *slices* (the symbol for a slice resembles a pizza slice).

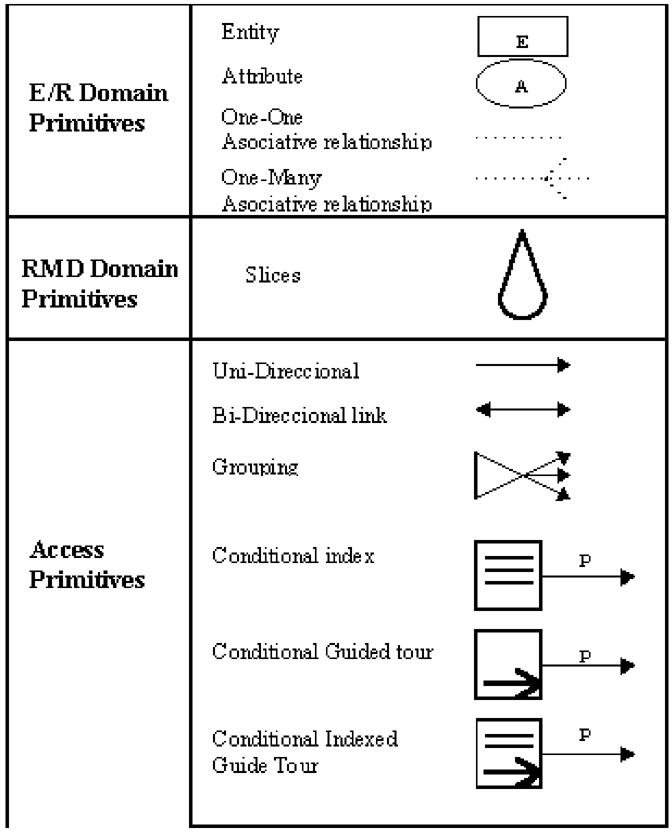


Figure 1: The elements of the RMM Data Model (RMDM)

RMDM specifies navigation via the six access primitives at the bottom of Figure 1. RMDM's most significant access structures are *indices, guided tours, indexed guided tours* and *groupings*. An index acts as a table of contents. A guided tour implements a linear path through a collection of items allowing the user to move forwards or backwards on the path. Indexed guided tours combine the functionality of indices and guided tours. Logical conditions qualify these access structures. For example, a condition "type=**panel**" attached to an index into a *conference_event* entity denotes an index to panels from that conference. The grouping mechanism serves as a major access gateway to other parts of the system, as often found on many applications' home pages or initial screens.

2.3 Extensions to RMM

Our experience enabled us to identify two useful additions to RMDM (1) *minimal slices,* and (2) *hybrid slices.* A **minimal slice** is a minimal collection of attributes of an entity type that enable users to uniquely identify entity instances. Although conceptually similar, RMDM minimal slices differ from database keys, because the latter usually do not convey relevant information (at a cognitive level) to users. For example, although an employee's social security number may serve as a database key, a minimal slice may instead comprise the employee's name and department. In hypermedia applications, minimal slices are used as anchors to entity instances.

Hybrid slices combine attributes from different entities and access structures. RMM's original slices are "pure" in that they combine attributes of the *same* entity type. Since every screen in an RMM application originates in a slice or in an access structure, the purity of slices severely restricts the allowable pages. For example, using pure slices, it is impossible to combine into one screen the name of a paper in a conference and an index of its authors. Hybrid slices can be used to that effect. We illustrate the usage of these two new kinds of slices in §5.3.1 (page 11).

3. Related Research

Hypermedia structuring and functionality requires design methodologies different from those developed for standard information systems. Hypermedia applications involve many different components, such as content preparation, structure, storage, user-interface, navigation, and retrieval. As a consequence, data models such as data flow diagrams, entity-relationship (E-R) diagrams and object-oriented hierarchies cannot represent the design intricacies hypermedia applications encompass.

Garzotto, Paolini, and Schwabe's HDM data model (1993) and its successor HDM2 (Garzotto et al., 1996) describes the structure of a database application domain adequately to support hypermedia access. HDM and HDM2 describe representation schemes, but provide little guidance on using those representations in the design process. In other words, while they describe an application domain; they do not constitute a hypermedia design and development methodology. RMM builds on HDM and HDM2 to provide this full methodology. Lange (1996) and Schwabe, Rossi and Barbosa (1996) have proposed hypermedia design methodologies based on the object oriented paradigm. For database domains, RMM has the advantage of using tools such as E-R diagrams, with which designers already are familiar. In fact, as we note in §5.9 designers using RMM especially have lauded this familiarity.

At the other extreme, Marshall and Shipman (1995) discuss authoring in spatially oriented hypermedia environments. In spatial hypertext, relationships may be left implicit, inferable only from the proximity of content with each other on the display. In addition, authors do not have to commit to a structure and its enforced consistency in advance. One could view this as an excuse to bypass consistency guidelines proposed by RMM and HDM; alternatively, time may show that the best spatial applications evolve to a state consistent with these guidelines. It is possible, in principle, to produce spatial designs from an underlying RMDM diagram. The interaction between the structured RMM methodology and spatial hypertexts remains to be explored.

Our experience follows Nanard et al., (1995) observation that hypermedia design "is an incremental and opportunistic human activity that takes place in a two axes space," in which one axis represents the technical aspect and the other axis represents the design process. Employing RMM certainly had both aspects. Nanard and Nanard also identified fast feedback loops as a fundamental requirement of a hypermedia development environment. As we shall describe starting in the following section, throughout LINKBase's design and development, we reiterated through design-implementation loops.

We also direct the reader to our prior research in systematic hypermedia user interface design (Balasubramanian and Turoff, 1995) and in automatically generating hypertext based on an application's internal structure (Bieber and Kimbrough, 1992; Bieber 1995). Neither has the breadth of RMM.

4. LINKBase

LINKBase is a new implementation of ACM SIGLINK's bibliographic application using information stored in a relational database management system. LINKBase contains information on *events* (conferences, workshops, special issue of journals, etc.), *event items* (actual articles), event *publication(s)* containing the articles (conference and workshop proceedings, books, URL addresses), the *organization(s)* sponsoring the event (e.g., ACM, INRIA, etc.), and *people* (authors, event managers, and organization officers). LINKBase is more than a database, it is a full-fledged hypermedia application which processes and reformulates data extracted from a relational database. The WWW was chosen as the delivery platform so the widest possible audience could access LINKBase easily and free-of-charge.

4.1 Using LINKBase

The following scenario illustrates LINKBase. A user interested in information about hypertext reaches ACM SIGLINK's LINKBase[2]. After traversing some introductory material, he or she selects "conferences" from a list of event types. The user arrives at the dynamically generated page shown in Figure 2a, which contains an index of hypertext related conferences. Upon selecting "ECHT '94" (European Conference on Hypermedia Technology 1994) the system generates the WWW page shown in Figure 2b, containing basic conference information including name, date, location, sponsor ("ACM"), and related publications ("ECHT '94 Proceedings"). Other navigation choices include "previous conference" and "next conference" for the index. Other links (not visible here) can lead to photographs taken at the conference, trip reports, a list of committee members, etc. At any time, the user has access to all major system entry points ("groupings") for events, authors, organizations and publications.

The user then traverses the link "Table of Contents" to the page shown in Figure 3a, containing a list of papers, posters and demos, etc., presented at ECHT '94 along with author names. Note that the "Navigation Path" reflects the user's movement deeper into the information space. Selecting "Adding Multimedia Collections to the Dexter Model," the user arrives at Figure 3b. From here the user can select author names and navigate to details about their affiliation and contact information.

4.2 Why use RMM for the re-implementation of LINKBase?

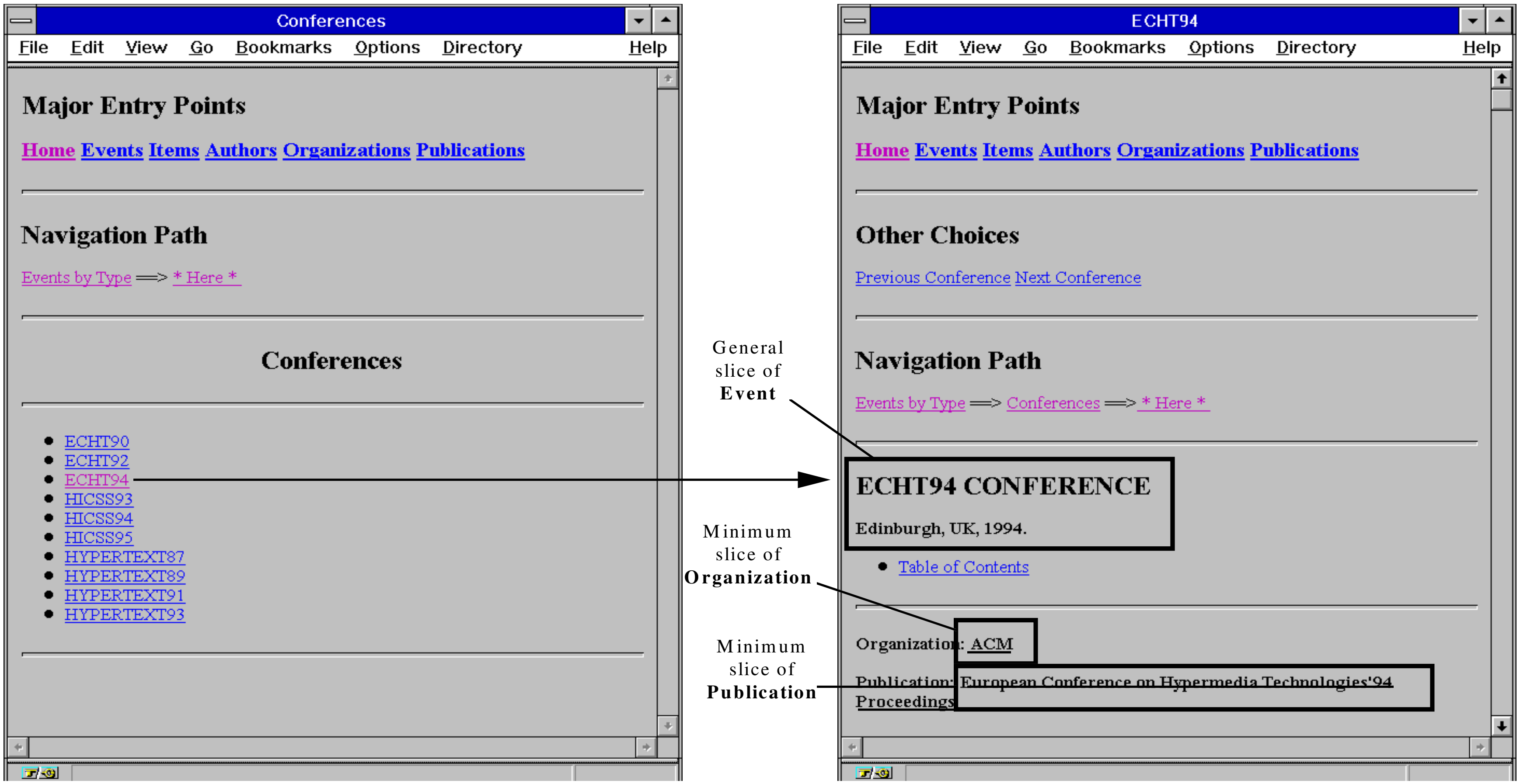
Like many Web applications, LINKBase's original version was designed in an *ad hoc* manner without following a methodology. In retrospect, this resulted in a system too limited in scope, containing only conference-related event items and co-author cross references. Source information was kept in flat files. The system administrator executed utilities to periodically update WWW pages in Hypertext Markup Language (HTML) and to re-generate navigation indices.

When it came time to upgrade LINKBase, the haphazard, semi-planned nature of the original system convinced us to use a systematic design methodology to facilitate a more comprehensive, consistent and robust product. We decided to use Ingres, a relational data base management system (RDBMS) as the back-end. Although industry-strength full-text retrieval engines such as WAIS™ [URL3] or BASIS WEBserver™ [URL4] or Topic® WebSearcher [URL5] successfully support good indexing and searching facilities on variable length text strings, they neither support the design process nor the management of relationships.

---

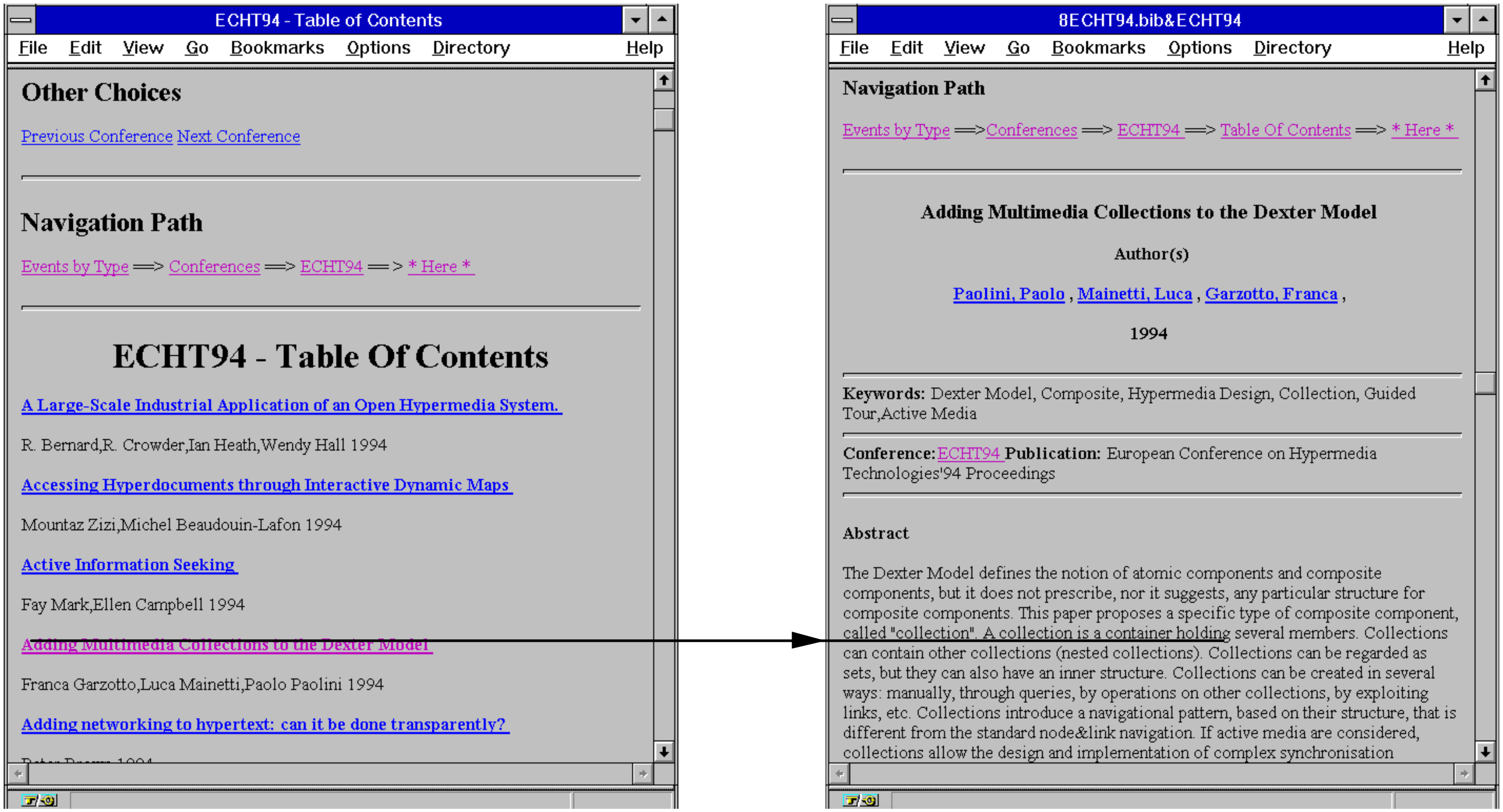
[2] Readers will be able to access LINKBase on the WWW through URL <http://www.acm.org/siglink/>.

Note to Reviewers: Due to changes in our computer systems, we currently are changing the platform on which LINKBase executes. We shall complete the conversion sometime during Summer 1996, until which time LINKBase (unfortunately and frustratingly) will be inaccessible. During our final testing of the conversion, reviewers will be able to access LINKBase from <http://space.njit.edu:5080/cacm/overview.html> by selecting the "LINKBase" button.



(a) (b)

Figure 2: Screenshots of a prototype LINKBase implementation showing navigation between dynamically generated Web pages. Screenshot (a) lists hypertext research conferences; (b) results from selecting "ECHT'94," and provides the conference's date and location, and access to its table of contents.



(a) (b)

Figure 3: Screenshot (a) shows the table of contents for ECHT '94 selected from Figure 2b. This lists all papers, posters, demonstrations and videos presented at the conference. When the user selects the paper "Adding Multimedia Collections to the Dexter Model," the system generates screenshot (b). It shows paper details such as authors, keywords, its abstract, etc.

LINKBase satisfied the criteria for an RMM-based design specified in (Isakowitz et al., 1995). The domain's structure contains no *ad hoc* relationships (except for our manually added introduction and acknowledgments, etc.—see §6.1.1), and can be expressed clearly and succinctly in terms of entities, relationships, and navigation structures based on these. Furthermore, we frequently add, and sometimes modify, information about authors and conferences, etc. While not a large system by industrial standards (Malcolm, Poltrock and Schuler 1991; Parunak, 1991), LINKBase does contain several hundred event items and authors. Although manual page generation would be possible, doing this in a controlled manner proves to be more efficient and robust. This also allows us to provide a controlled interface for updates (see §6.1.1 on page 22) instead of letting (numerous) people update HTML pages at will and potentially degrading the consistent layout and integrity of LINKBase.

5. LINKBase Case Study

This section describes LINKBase's design and development using RMM.

5.1 Requirements Analysis

Through interviews with users we arrived at the following major criteria for LINKBase's reimplementation :

(1) Incorporate information missing from the original LINKBase system.

(2) Use a common World-Wide Web browser as the interface, so our users (academics, researchers and graduate students around the world) have easy access to the system.

(3) Store the information in a database for flexible access and easy update.

(4) Automatically generate the HTML pages from the database contents so no one updates pages manually, and so no one has to write HTML manually.

(5) HTML pages need to be generated at run-time (as opposed to periodically updating them), since we have no control over when people update the system and it always needs to be up-to-date.

(6) Much of the system content would be incomplete. For example, the authors themselves would have to sign onto the system to add and update address information only after their articles were entered into the system. Practically, it could be quite a while, if ever, until they got around to this.

We determined the final set of entities in an iterative fashion. When we wrote the requirements documents, we only planned to add author contact information. But as we thought about the two main entities: authors and conference proceedings articles, we realized that we could provide users with additional useful information. Discussing the system with prospective users resulted in further entities and relationships.

5.2 Entity-Relationship Design

The first step, *entity-relationship (E-R) design,* involves identifying the entities and relationships that make up the application domain. These entities and relationships will become nodes and links in the resulting hypermedia application. This step is familiar to most systems analysts with experience in developing applications employing relational databases. It results in an E-R diagram for the entire application.

Entity/Relationship	Attributes

Organization	OrgName, OrgType (university, SIG, professional organization, etc.), Address, Phone, Fax, Email, Homepage, MiscInfo
Event	EventName, EventType (conference, workshop, book, special issue), Series Name, Date, Location, Description, List of Referees, Trip Report, Event Photo, MiscInfo
Event_item	EventItemTitle, EventItemType (introduction, paper, panel, video, demo, poster, workshop_related, technical briefing, cultural briefing, commercial symposium, tutorial, keynote speech), Category or Grouping within Event, Subject Classification, Abstract, Keywords, Content, Full reference, Page #, MiscInfo
Person	PersonLastName, PersonFirstName, Affiliation, Address, Phone, Fax, Email, Biography, Homepage, MiscInfo
Publication	PublicationName, Publisher Name, Location, ISBN, Foreword, MiscInfo
Organizes	OrgName, EventName, EventType
Manages	PersonLastName, PersonFirstName, EventName, EventType
Produces	PersonLastName, PersonFirstName, EventItemTitle
Edits	PersonLastName, PersonFirstName, PublicationName
Officer_of	PersonLastName, PersonFirstName, OrgName
Belongs_to	PublicationName, EventName, EventType
Consists_of	EventName, EventType, EventItemTitle
Contains	PublicationName, EventItemTitle

Table 1: Entity and relationship attributes in LINKBase's domain.

5.2.1 LINKBase Example

In our scenario the user was interested in aspects of the ECHT'94 conference, such as its location, sponsoring organization, proceedings, committee members, any trip reports, and details of items presented (papers, posters, demonstrations, videos, etc.). Similarly, a user looking at a paper abstract may be interested in more details about it, background information about its authors, and general information about the conference at which it was presented.

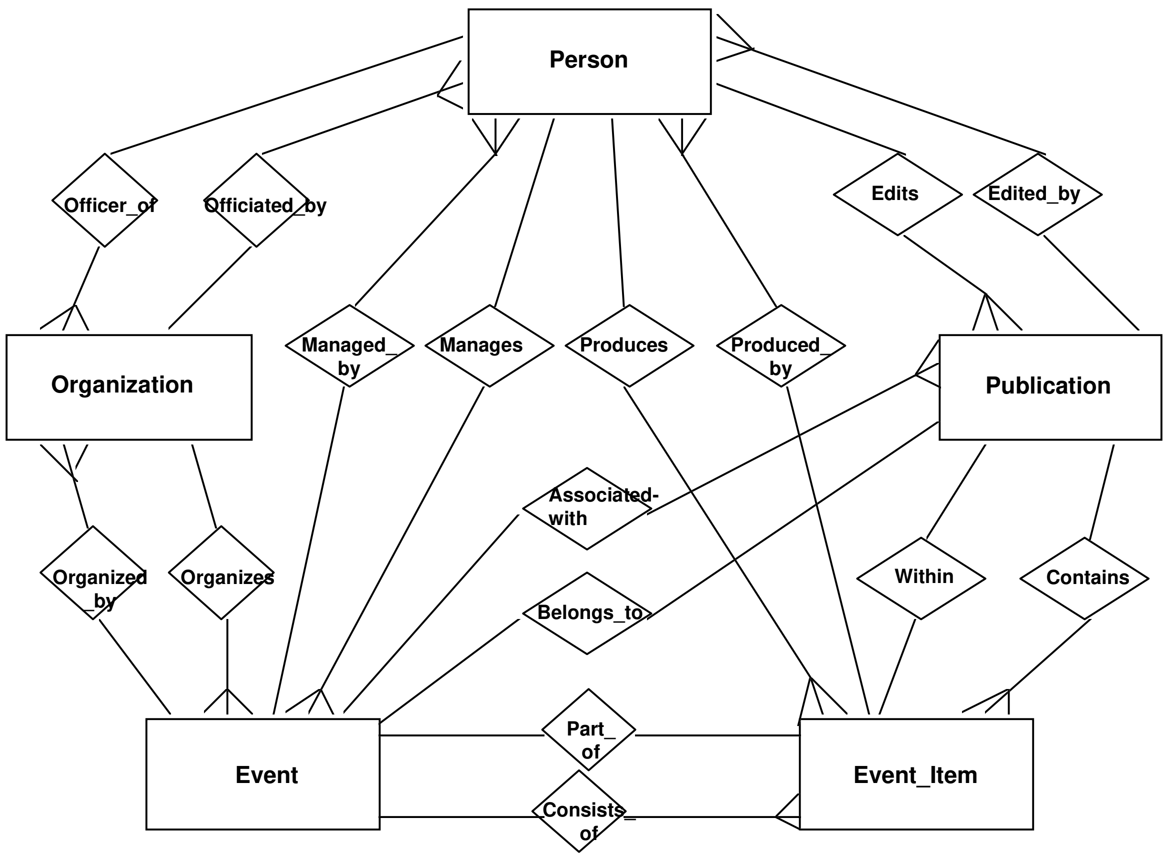


Figure 4: E-R Diagram for LINKBase application.

Based on these requirements, we identified five *entities* **Organization**, **Event**, **Event_item**, **Person**, and **Publication** and eight associative relationships[3]. Figure 4 shows our E-R diagram. Table 1 lists the entity attributes. Each entity has a general "catch all" attribute *MiscInfo* for including additional information at a later date. A person can be an officer of many organizations. Conversely, an organization can have many officers. Thus the relationships **Officer_of** and **Officiated_by** relate the **Person** and **Organization** entities. These are the sole time-sensitive relationships in our system, as they reflect an organization's current status. In our bibliographic domain, information is mostly of archival nature. An organization can organize many events (both simultaneously and over time). Conversely, organizations can jointly-sponsor an event. Thus relationships **Organizes** and **Organized_by** relate **Organization** and **Event**. Many people manage an event and, over time, a person can manage many events. The remaining relationships in Figure 4 follow a similar pattern.

5.2.2 Lessons Learned

During the E-R design step, we learned the importance of thinking about entities from the point of view of their interrelationships, and how much information a user possibly could want to access. This led us to include additional attributes and additional entities we did not anticipate originally. This user-access focus also prompted us to add additional direct navigational

---

[3] In this section, for distinction we write entity and relationship names in boldface and entity attributes in italics. In later sections it suffices to write both in italics. Also note that relationships are shown in both directions.

relationships when the granularity or units of information would not give users direct access to what we believed they wanted.

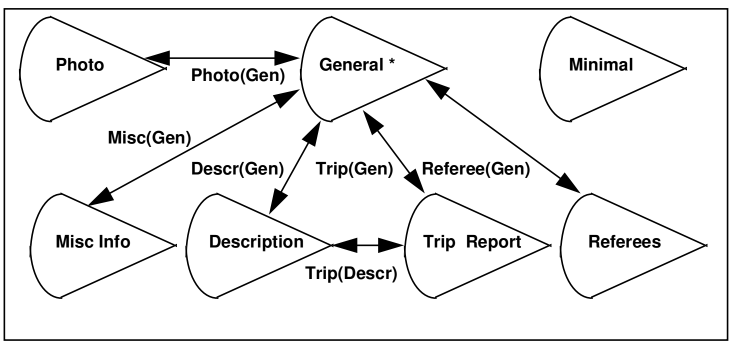
5.3 Slice Design

The second step, *slice design*, involves grouping entity attributes for display. This task is unique to hypermedia applications. It involves *slicing* an entity's set of attributes into different overlapping but meaningful subsets, displayed one per page. A *head* slice is the default target of access structures entering an entity; it usually connects to all other slices via *structural* links, which carry semantic labels and may be uni-directional or bi-directional. Labeling the links well highlights the nature of the target slices. As described in section 2.3, designing LINKBase led us to extend RMM's data model by introducing *minimal* and *hybrid* slices.

5.3.1 LINKBase Example

Since they are unique contributions of this case study, we describe how minimal and hybrid slices were used. Figure 5 shows the slice diagram for the **Event** entity. Its **minimal slice** contains two attributes: ***EventName*** and ***EventType***, for example, "ECHT'94 Conference", which, for most purposes, suffice to identify an event. The six regular slices contain (1) general information, (2) a description, (3) trip reports, (4) photographs, (5) a list of referees, and (6) miscellaneous information. The *general* slice, which contains basic information such as the entity's name, type, series name (e.g., ACM hypertext conferences), date and location, is the head slice.

Event



General * : EventName, EventType, SeriesName, Date, Location

Minimal: EventName, EventType

Description: EventName, DescriptionText

Trip Report: EventName, TripReportText

Referees: EventName, List of Names

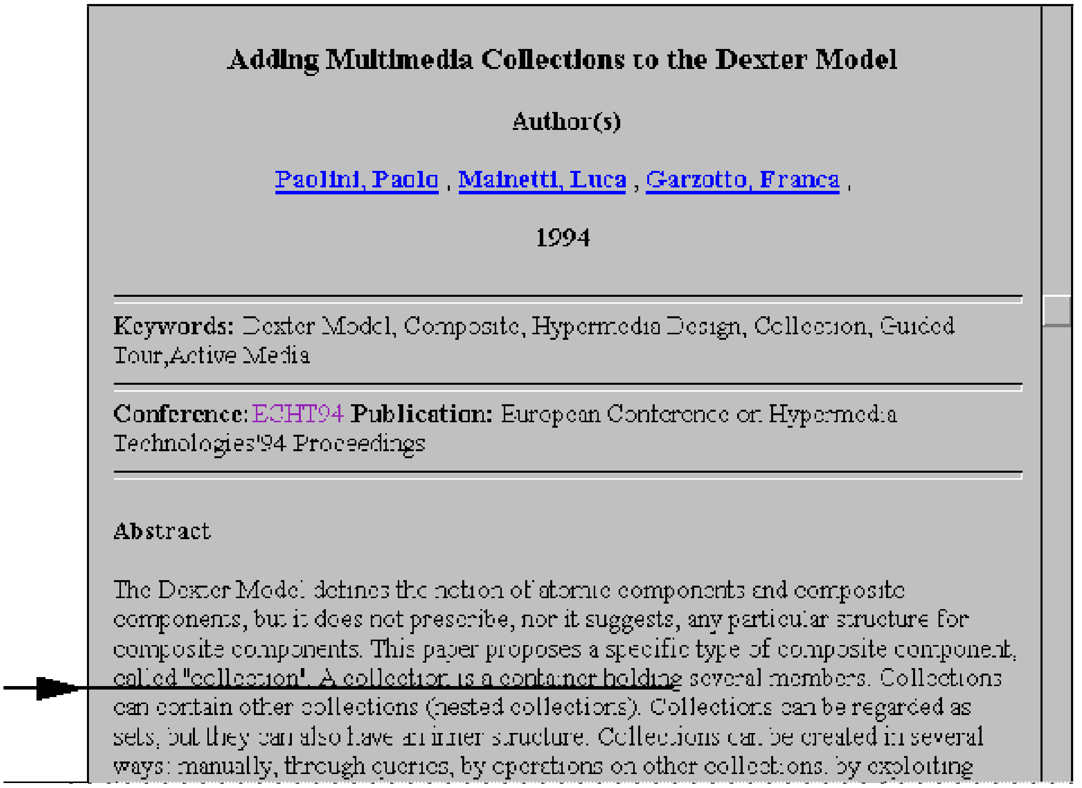
Photo: EventName, EventPhoto

MiscInfo: EventName, MiscInfo Text

Figure 5: Slice diagram for the Event entity showing its slices and their attributes. The general slice serves as the head slice. The minimal slice contains the minimum set of context information necessary to identify this entity when embedding it within hybrid slices of other entities.

Displaying information about **Event_Item** illustrates the use **of hybrid slices.** Figure 6 shows an instance of an event item as seen on a Netscape browser. Its schematic screen, as shown in Figure 7, incorporates attributes of different entities. **Event_Item ->** ***General***, for example, refers to the **General** slice of the **Event_Item** entity, while **{Produced_by} Person ->** ***Minimal***, refers to the minimal slice of the **Person** entity related to the **Event_Item** via the **Produced_by** relationship from Figure 4's E-R diagram. More generally, the format is: "{relation} entity -> *slice name.*"

A hybrid slice is conceptualized as a composition of slices (pure or hybrid) from other entities, that is anchored in one specific entity . Figure 8 shows the design of the hybrid slice we have been discussing. It is anchored in **Event**_Item. Figure 8 illustrates how its design combines slices from the **Person, Event_Item, Event** and **Publication** entities.



Adding Multimedia Collections to the Dexter Model

Author(s)

Paolini, Paolo , Mainetti, Luca , Garzotto, Franca ,

1994

Keywords: Dexter Model, Composite, Hypermedia Design, Collection, Guided Tour,Active Media

Conference: ECHT94 **Publication:** European Conference on Hypermedia Technologies'94 Proceedings

Abstract

The Dexter Model defines the notion of atomic components and composite components, but it does not prescribe, nor it suggests, any particular structure for composite components. This paper proposes a specific type of composite component, called "collection". A collection is a container holding several members. Collections can contain other collections (nested collections). Collections can be regarded as sets, but they can also have an inner structure. Collections can be created in several ways: manually, through queries, by operations on other collections, by exploiting

Figure 6: An instance of a hybrid slice as viewed in a Netscape browser

Event_item -> *General*

{Produced_by} Person -> *Minimum*

{Part_of} Event -> *Minimum*

{Within} Publication -> *Minimum*

Event_item -> *Content*

Event_item -> *Reference Information*

Event_item -> *Miscellaneous Information*

Figure 7: The hybrid slice definition for Event_item. It includes slices from the **Person, Event_Item, Event** and **Publication** entities in the format: "{relation} entity -> *slice name*."

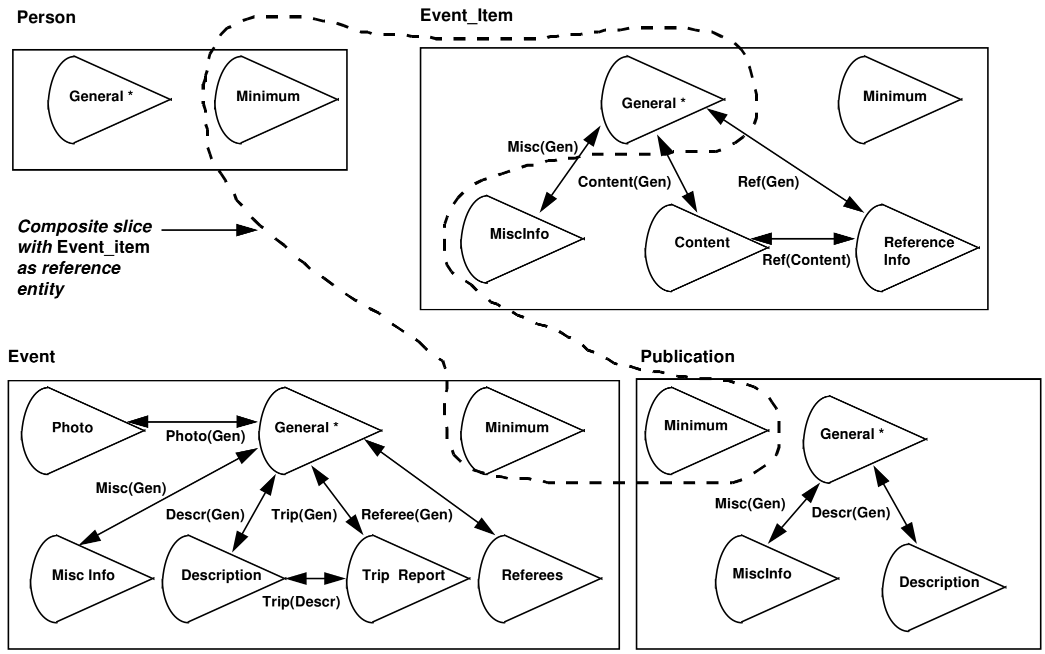


Figure 8: The general slice of Event_item and the minimum slices of Event, Person, and Publication have been combined together (covered by the dashed lines) to form Figure 6 andFigure 7's hybrid slice representing an Event_item.

5.3.2 Lessons Learned

During slice design the designer begins to refine and prune the application's design. For example, one might decide to drop various attributes included in the E-R diagram, which, in retrospect, proved unnecessary or impractical. For example, we dropped the *biography* attribute from **Person** and *subject classification* from **Event_item**, as both would turn out to be impractical to collect and maintain.

In our application, we utilized a single hybrid slice per entity, which became that entity's head slice. The essence of slice design lies in ranking information according to its importance and clustering information for its cohesiveness. Slice design involves several calculated tradeoffs. Small (screen-sized) chunks eliminate scrolling (meaning all information is immediately visible), but require more separate slices, thus potentially increasing fragmentation and reducing local coherence (Kahn, 1995; Thüring, Hannemann and Haake 1995). This tradeoff is the subject of a long-standing philosophical and user interface-oriented debate in the hypermedia community (Akscyn, McCracken and Yoder 1988; Conklin, 1987; Raskin, 1987). Although interface considerations come later in RMM, we needed to know the approximate size of our windows to determine how much we could display without scrolling (but without overloading the user by having too much visible at once). On the other hand, knowing that navigation on our implementation vehicle—the WWW—is often slow, pragmatically we may wish to have fewer, fuller nodes. We also might place more attributes in minimal slices to reduce the need to traverse to other entities for related information.

5.4 Navigation Design

The third step, *navigation design*, involves identifying the paths that enable hypertext navigation. We analyze each *associative* relationship between entities and each *structural* relationship between slices to determine first whether to include it in the final system, and second, which kind of access to implement for it. We replace each by one of Figure 1's (page 3) RMDM access structure. Because we dynamically generate all displays, we can include no hard-wired links between particular instances. Instead we specify links based on entity attributes and relationships. For entities with links leading to multiple instances, we employ conditional indexes, conditional guided tours, or conditional indexed guided tours. When links lead to single instances—the norm for structural links—then the RMDM unidirectional access mechanism suffices. This step produces an *RMD diagram*.

5.4.1 LINKBase Example

Figure 9 shows LINKBase's RMD diagram. The RMD diagram shows only associative relationships at the entity level. We provided the following major entry points: through **Event** indexes by series name and by type, **Event_item** indexes by subject and by assigned keyword, an **Author** index, an **Organization** index, and a **Publication** index. For example, to see the papers (event items) from the ECHT'94 conference, a reader would follow this navigation path: First the user chooses the **Event** index by type ("type = 'conference'"). From the list of conferences, the user selects a particular instance, "ECHT'94," producing all papers presented there (**Event_item** index by type; "type = 'papers'"). Each **Event_item** index comprises a table of contents of all papers, posters or demonstrations, etc., satisfying the condition specified. Attribute values serve as the conditions. The system generates each of the six **Event_item** indexes dynamically based on the selection criteria (by name, by type, by grouping, by subject). We chose conditional indexed guided tours for each. Once viewing a particular instance of **Event_item**, the reader can navigate directly to the next and previous instance in the guided tour. In our example, when viewing an ECHT'94 paper, the reader can travel to the next and previous entries in that conference's list of papers. The reader can access detailed information (beyond what the minimal slice contains) such as its author entity instances, the associated publication entity instances, etc., through the associative links.

Normally one would provide major entry points (in the form of groupings) to all major entities. In our case, knowledge about how readers would employ LINKBase led us to provide only a "conditional" grouping for the **People** entity, restricting major access to only authors.

5.4.2 Lessons Learned

RMM enabled us to plan our access structures only logically in this step, postponing all presentation details to user interface design (step 5). We mostly used indexes and indexed guided tours. We did not use simple guided tours anywhere because we anticipated the number of instances of each entity to be relatively high. Guided tours without an index suffice only when an entity has relatively few instances. Indexes allow more direct access and should be used for a relatively high number of instances. Indexed guided tours serve best in-between.

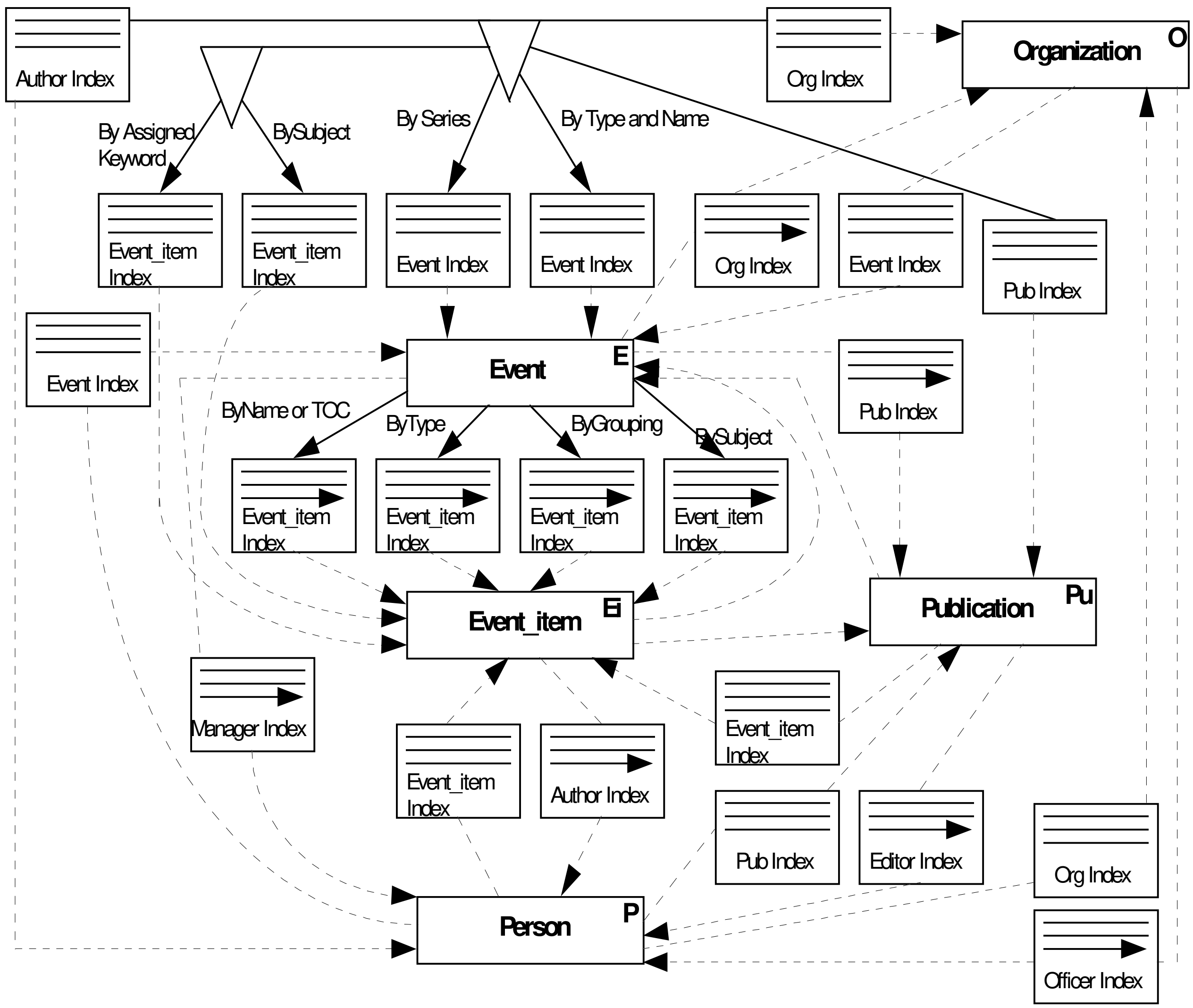


Figure 9: RMD Diagram for LINKBase.

5.5 Conversion Protocol Design

Isakowitz, et al.,(1995) do not elaborate on the remaining four steps of RMM. In the next four subsections we discuss our approach. While the first three steps can be considered logical design from a software development life-cycle perspective, the remaining four steps are more related to physical design, that is, translating the conceptual model into a working artifact.

The fourth step, *conversion protocol design*, produces a set of rules (to be implemented as a program or as instructions to programmers) for converting components of RMD and E-R diagrams into physical objects in the target hypermedia application. This includes translating the logical E-R design to a physical design in terms of relational database tables, columns and keys. Entities become tables. Attributes become table columns. Indexes are manifested as lists of database query results.

5.5.1 LINKBase Example

We converted the logical E-R diagram to physical tables on Ingres and transformed indexes in the RMD diagram into queries that, upon activation, produced a list of links to instances. We also identified attributes that could be null given the fact that we would populate the system initially from an existing source without all the attributes of the model's entities (e.g., without author addresses). For each we needed to decide whether to display "none" as its value or not to display its label at all. Displaying "none" is more consistent for readers who may expect a value and wonder at its absence. When readers do not expect certain information, we can omit its label when valueless.

To reduce the size of database records, we also decided to store each event item's abstract as a file in the UNIX file system and the filename as the *abstract* attribute's value. Because **Event_item** did not have a unique identifier, we introduced the attribute *Event_Item_Id* . Its value is generated as each instance of **Event_item** is inserted. It subsequently could be used internally for retrieval (or as part of the abstract's address).

5.5.2 Lessons Learned

In order to improve retrieval efficiency, one may have to "de-normalize" the normalized data model. Of course, one should consider the implications of increased redundancy. The tradeoff is between retrieving long rows of data versus joining tables and retrieving related items. This, in turn, depends on whether one plans to convert application objects dynamically at run time as we planned, or in advance, as with departmental information system illustrated by Isakowitz, et al., (1995). Design changes often must be made in order to convert the schema into tables, columns, keys and indexes. Items without an inherent unique attribute, must be assigned a unique identifier if instances are to be retrieved individually.

5.6 Step 5: User Interface Design

The fifth step, *user interface design*, involves designing screen layouts for every RMD diagram object. We focus on four important aspects: the information content of slices and windows identified in the second step; access structures to linked information from the third step; general navigational aids for backtracking and overview displays; and the layout of link anchors, how nodes and indexes appear, and the location of navigational aids.

Just like any other application, one could employ low-fidelity prototyping using paper and pencil strategies to prototype the interface (Rettig, 1994) or employ high-fidelity prototyping with computerized tools (Winograd, 1995).

5.6.1 LINKBase Example

During this step, we sketched each major window (one per slice or index) and the navigation pathways between the windows. Figure 10 and Figure 11 give examples. We made changes to our navigational access design after a few iterations. For example, we converted the Event Index by *type* and *name* from indexes (as shown in Figure 9's RMD diagram) into indexed guided tours, as Figure 2b and Figure 10b reflect.

In order to understand the flow of HTML pages and various navigation pathways better, we constructed some sample pages by hand, linked them together, and tested them using a Web browser. Each window, from a graphical user interface point of view, became a HTML page in our application. Thus, we adopted both static (paper) and dynamic (working) prototyping techniques.

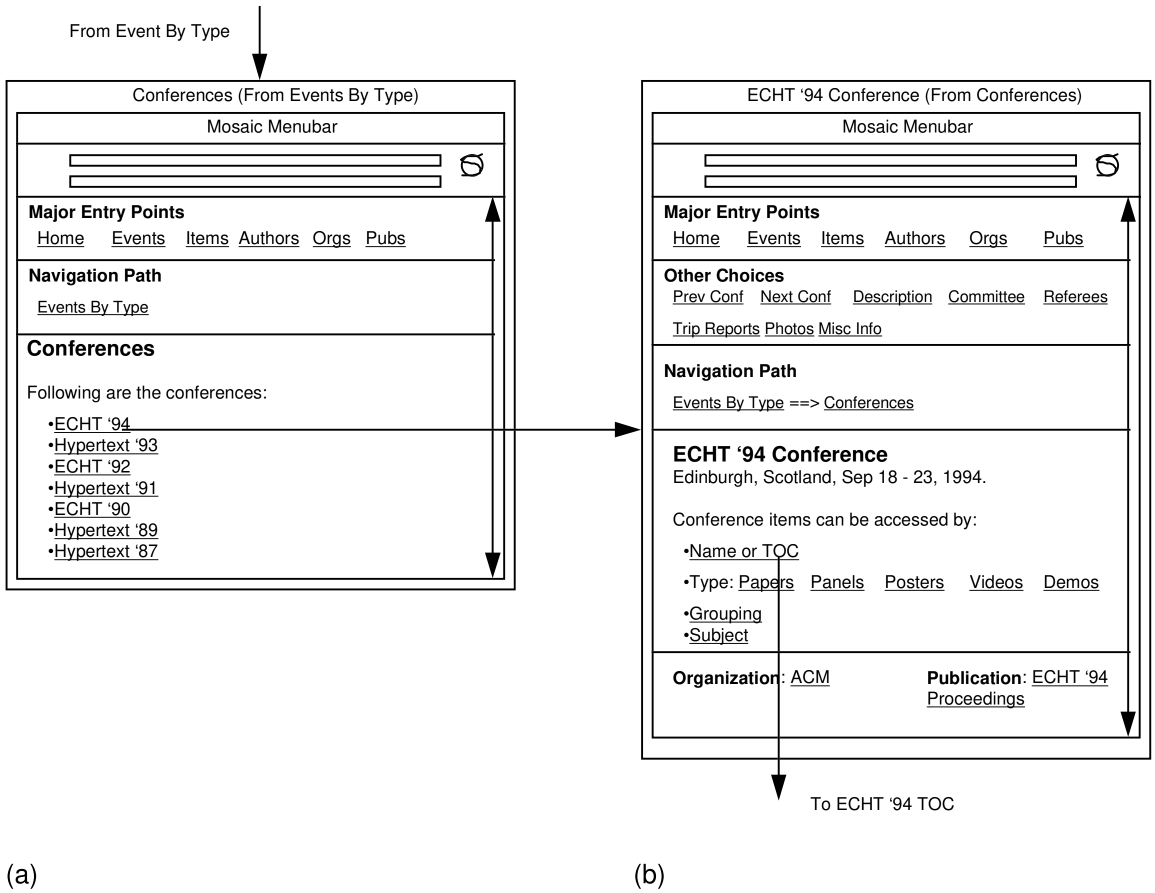


Figure 10: User interface design and navigation pathways: Low-fidelity sketches of our windows. The window with a list of conferences in (a) is generated from the "Homepage" upon selecting "Events By Type" and then "Conferences." It prototypes the general layout for an index. Clicking on a conference name generates the window shown in (b), which portrays the general layout for an **Event** entity. It contains structural links to other slices (under "other choices") and associate links to indexes of this conference's **Event_items**.

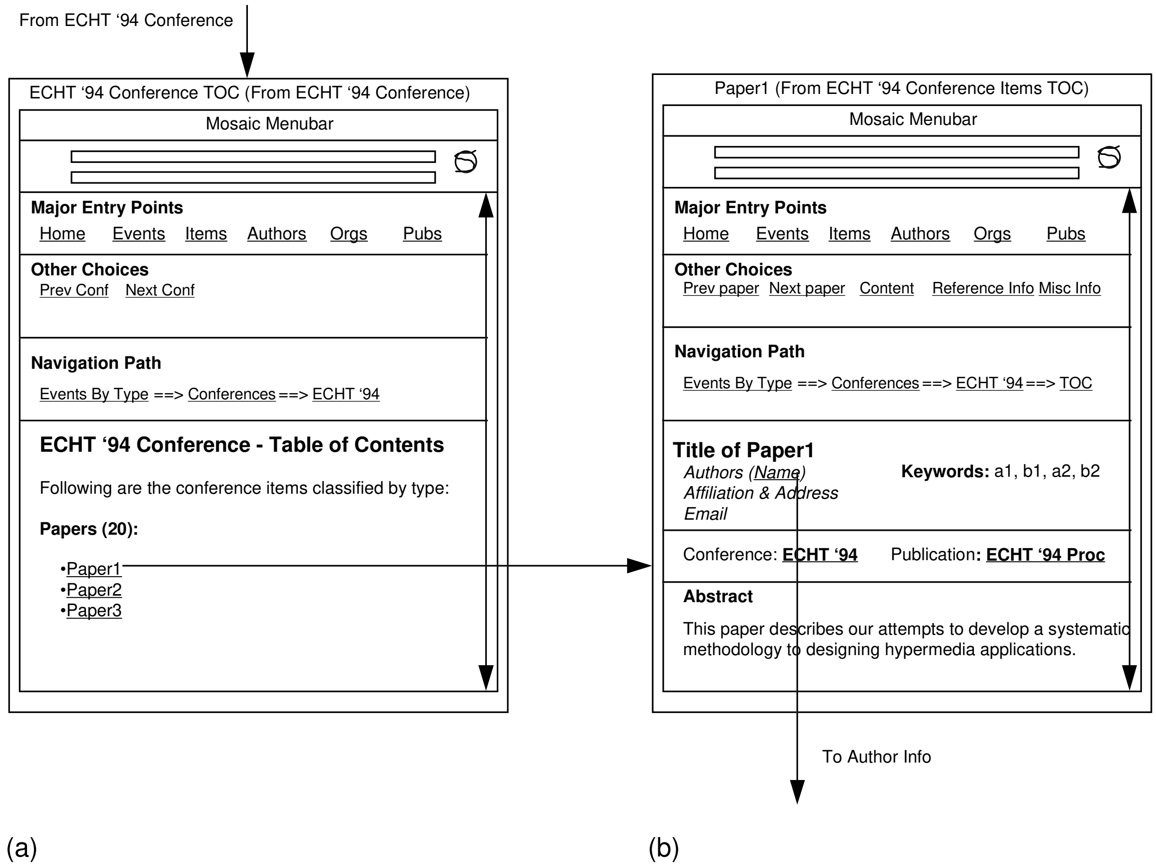


Figure 11: User interface design and navigation pathways, continued. The window in (a) prototypes the general layout of an indexed guided tour's leading index page. It is generated by selecting "Table of Contents" in Figure 10b. It shows the **Event_items** grouped by type such as papers, posters, etc. The window in (b) portrays a page along the indexed guided tour (with links to the previous and next papers under "other choices"), generated upon selecting a paper in (a) or a previous/next tour link. This page implements the general slice node design for an **Event_item** from Figure 7 for an article. The navigation path in each window reflects the reader's traversals.

We adopted many user interface design guidelines suggested by hypertext researchers. To maintain a sense of orientation and facilitate navigation, we followed the guidelines for local and global context by Kahn (1995) and Thüring et al. (1995). For example, users may have difficulty distinguishing between structural links within an entity and associative links to other entities, as WWW browsers display all anchors in the same format. The developer must provide semantic cues through the surrounding layout and through good labeling. We employ a stable screen layout, as prescribed by Thüring et al.'s (1995) eighth hypermedia design principle. We placed global links under the header "Major Entry Points," which contains all the major groupings from Figure 9's RMD diagram. Major entry points provide the reader with *landmarks* (Nielsen 1995) or *strategic choices* (Balasubramanian et al. 1995), which always are visible. To minimize clutter and scrolling, and hence *user interface adjustment* (Thüring et al., 1995), we placed all structural link anchors to other slices under the heading "Other Choices." We represented associative links and all other access structures as anchors within the window's main content area. Although not implemented in LINKBase, a modified version of the RMD diagram (implemented as a conditional image map) could serve as an overview diagram providing global orientation and high-level navigation. See [URL8] for an example.

We planned to represent link anchors in graphical or iconic form, especially for the major groupings. However, we quickly decided against this due to the complexity of icon design and usability testing (Nielsen and Sano 1994). Instead, we chose simple textual link anchors.

5.6.2 Lessons Learned

Because Web browsers at the time did not offer frames, and we chose to use a single window per slice, we had to use the window space very judiciously. Our design evolved the following conventions. We grouped all important information to the top of each page. We clustered all related items together to preserve context. We tried to position no cluster of information below the visible portion of a window when opened, so readers would always know what kind of information a window contained and what kind of information they would find upon scrolling.

In LINKBase we implemented each index in a separate window from the entity slices accessible through major groupings or associative links. In a separate RMM design project for a university information system, we found it often made sense to embed some indexes within the slices from which they would be accessed. In these cases the slice had few attributes, so the index was visible without scrolling, and the index had few entries. For example, in the general slice for a course section we included a list of that section's instructors. This list represented the *taught_by* index, which often contained a single instance, but when team taught included multiple instructors and, when available, included the section's teaching assistant.

5.7 Step 6: Runtime Behavior Design

In the sixth step, *runtime behavior design*, one designs the programs that will control how your application generates and retrieves information, and how it interacts with the browser and the user. One can design HTML production dynamically (generated at run-time in response to user interaction) or in batch-mode (periodically generating all pages for future "static" retrieval). RMM supports both. Specifying runtime behavior also involves designing the algorithms and implementation mechanisms for hypertext navigation such as browsing (link traversal), history tracking, indexed navigation, guided tours and backtracking (such as from our navigation path feature). The designer should identify those parts of the information content that need to be generated dynamically, which parts constitute link anchors, to what their links lead, and each link's dynamics—the "behavioral semantics" or the programs to invoke when the user selects each anchor. This in turn will dictate the parameters and URL destination to embed within the HTML anchor. (This is similar to the object-oriented concept of attaching methods to data elements.) The designer determines all these aspects based on the user interface design sketches for each node, which in turn rely on the RMD diagram and the slice design diagrams.

5.7.1 LINKBase Example

This step required detailed design of the interaction between the WWW server and the Ingres database server through the WWW's Common Gateway Interface (CGI). CGI is the mechanism for communicating between an external information source such as a database and the Web server (Liu, Peek, Jones, Buus and Nye 1994). The next section presents details of this interaction.

To preserve both local context and current position (design principles three and six by Thüring, et al., (1995)), we added the "Navigation Path," which the system dynamically generates based on the user's link traversals. For example, the table of contents window in Figure 8a has the navigation path "Events By Type ==> Conferences ==> ECHT '94." The current implementation appends "-> *here*" to the navigation path because some early users were confused as to whether the last page listed was the current one visible. This required us to maintain state information as the user navigated through the information space. Web browsers and servers are stateless (Berners-Lee, Cailliau, Luotonen, Nielsen and Secret 1994), so we pass state information around through the CGI environment variable QUERY_STRING. Of course this state information is lost if the user navigates out of and reenters LINKBase (without backtracking). State information can now be preserved, to some extent, using the "cookies" mechanism in Netscape.

5.7.2 Lessons Learned

We had to iterate designing the runtime behavior with construction and testing (step 7) so that we could observe the dynamic behavior of our C programs and argument passing through the CGI. Most of our lessons learned were technical. Since this was the first time we interacted with a database through CGI, we had to figure out the mechanics of coordinating the Web server and an Ingres database server through CGI. Apparently the details for each relational database server's CGI gateway is different. We especially relied on Liu et al.'s book (1994) and on Web resources concerning the CGI [URL6].

5.8 Construction and Testing

The final step, *construction and testing*, implements the design obtained through the first six steps. It involves constructing the physical database out of the logical design (developed during Step 1 and turned physical in Step 4), populating it with domain data and implementing each of the mechanisms designed in the previous step. After construction, one should test the application to ensure that it satisfies functional, navigational and usability requirements. For hypermedia applications this includes typing each link to ensure we generate the proper underlying parameters.

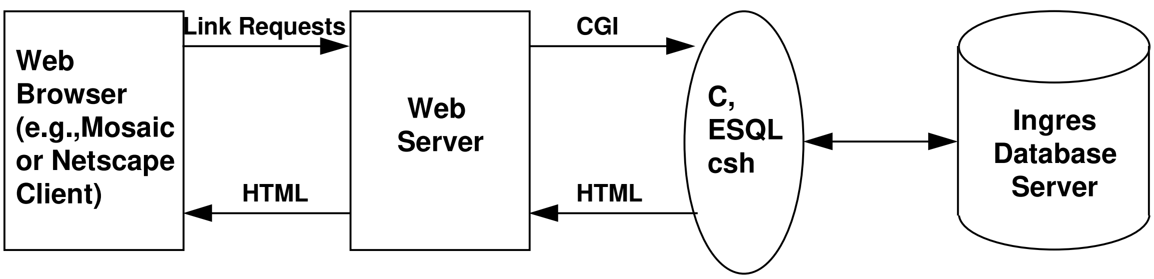


Figure 12: LINKBase's Client/Server Architecture

5.8.1 LINKBase Example

We created a database on Ingres with tables for the five entities and their associative relationships. We populated the database using C programs, extracting information about all ACM sponsored events in the field of hypertext/hypermedia from the HCI Bibliography Project files [URL7]. We had to update some data manually, such as for **Organization** (all attributes) and **Person** (e.g., address and email). We placed the text of abstracts in UNIX files.

Figure 12 shows the client-server architecture we adopted. This Web-database gateway approach is standard for any Web application that uses information from a dynamic source. The system functions as follows:

1. The Web browser detects when the reader selects a link and passes it to the Web server. For example, consider Figure 2: Screenshots of a prototype LINKBase implementation showing navigation between dynamically generated Web pages. Screenshot (a) lists hypertext research conferences; (b) results from selecting "ECHT'94," and provides the conference's date and location, and access to its table of contents. and Figure 10b. Assume the user requests a list of all papers (**Event_items** of type "paper") associated with ECHT'94 (an **Event** of type "conference").

2. The Web server invokes a *csh* script, residing in the *cgi-bin/bin* directory. The script invokes a C program, passing the arguments “ECHT’94” and “paper” through the CGI environment variable QUERY_STRING. The program in turn passes these arguments as variables to embedded SQL (ESQL) statements.
3. These SQL queries execute over the database. The Ingres server returns results (rows of paper titles and author names).
4. The C program converts the results into HTML. That is, it adds appropriate HTML tags around the result set. Appropriate portions of the result’s text are replaced by HTML anchors containing URLs and arguments for the corresponding C programs to invoke if selected by the reader.
5. The Web server is then informed that the stream coming through the standard output is a HTML document, that is, the “content-type” is “text/html.” The Web server forwards the HTML document to the client Web browser for display.

In this way, LINKBase dynamically generates each HTML document (with a few exceptions covered in §6.1.1) based on embedded SQL queries.

We treated each interaction as a separate database session, opening and closing the connection, because we could not tell when a user navigates out of LINKBase. We are investigating a time-out mechanism to keep the database session open for a certain period of time, as users probably will navigate within our information space for a while.

5.8.2 Lessons Learned

It became clear that runtime behavior design, and construction and testing (steps 6 and 7) go hand-in-hand. We started implementation with a small subset of the system. As we gained better understanding of the initial implementation related issues, we slowly extended the prototype to other relationships, navigational mechanisms and views. This experience refined our own understanding and belief in RMM’s systematic approach guiding LINKBase’s development.

5.9 Developer Reactions to RMM

Both RMM developers (of steps 1-4 or 1-5) and implementors of RMM designs have praised the methodology. LINKBase's primary developer, who is an author of this paper, notes:

"I felt the methodology was extremely useful and relevant, as well as easy and intuitive to use. It was a natural extension to the design approaches I am used to in the relational world. Something like OOHDM [Object-Oriented Hypermedia Design Methodology (Schwabe et al. 1996)] would have been a major learning curve since I do not know much about OMT or object-oriented methodologies. Imposing relationships between entities to facilitate navigation (though they may not be related in the real world) is a powerful feature of RMM. The RMD diagram is another powerful feature. This was the first time I have ever worked with a formal way of representing navigation and access structures for an application. I wish there were something like this for non-hypermedia applications also."

The RMM developer for the university information system mentioned in this paper also is very enthusiastic about RMM. He found it "easy because I already know database concepts. No parts [steps] of the design were difficult, and at each part we only dealt with a small part of the design. The end result looks difficult, but it was not complex at all [to design any particular part]... [RMM] was perfect."

LINKBase's RMM implementor commented that the RMM outputs made it easy to visualize the system, its structure and what to place on each page. Learning about the system was made "really easy" due to the expressiveness of the RMM output. The RMM implementor on the university information system agreed. The resulting RMM design was "intuitive" and he implemented it "almost unconsciously." However, consistent with the limitations noted in §6.1.1, the first implementor was frustrated that RMM provides so little help on implementing the design.

6. ISSUES IN SYSTEMATIC HYPERMEDIA DESIGN

Despite the abundance of WWW applications, to date neither the research nor the professional literature has reported on how one should design them systematically. Through this case study, we have successfully demonstrated that systematic design methodologies can be applied to Web-based applications, including those that generate information dynamically. We can apply our methodology to new hypermedia applications as well as retrofitting existing legacy applications with a hypermedia interface (§6.3). RMM can be used to define the interface of any domain that either has a single entity with numerous attributes (requiring slice design and structural linking) or multiple interrelated entities with multiple instances, in which case users benefit from directly accessing the associative links. Practitioners embarking on major information service initiatives should find such a systematic methodology of great value.

While Isakowitz et al., (1995) lay the foundation for the seven-step RMM, they provide details only for the first three steps. This article contributes by expanding on the remaining four. During entity design we added three intermediate steps of identifying minimum slices, hybrid slices and node design. We also contributed the concept of making the logical data model physical during conversion protocol design.

With the case study as background, in the following subsections we broaden our discussion of RMM and hypermedia design beyond the LINKBase application. We begin in §6.1 by considering RMM's strengths and limitations, and by proposing some extensions. In §6.2 we expand on performing a hypertext requirements analysis, which should precede any RMM analysis. In §6.3 we consider applying RMM to an organization's existing legacy systems. We close in §6.4 with some general design lessons we have learned.

6.1 RMM: Strengths, Limitations and Extensions

With RMM the designer need not learn an entirely new way of designing hypermedia applications. RMM builds upon and thus adds value to existing ways of designing and developing applications. For example, the designer still works with an E-R design to model real-world objects and relationships, but also analyzes the application domain from a hypertext philosophy of considering all possible (internal and external) relationships and maximum user access (see §6.2). This led us, for example, not only to include additional entities and attributes, but also to capture additional relationships facilitating navigation (e.g., for the sake of adding a navigational relationship, identifying an **Event_item** as **within** a **publication)**. The power of RMM also lies in the fact that the E-R model is directly reflected at the presentation layer thereby reducing both functional opacity (mismatch between the reader's mental model and the interface) (Rao and Turoff, 1990) and system opacity (mismatch between the implementation model and the interface) (Brown and Newman, 1985).

6.1.1 *RMM Limitations and Extensions*

Whereas RMM focuses on the logical design of a hypermedia application, it does not offer details about physical design or implementation. The actual physical design and development is left to two steps, conversion protocol design and construction and testing, which need greater elaboration. For example, it is not clear whether normalization benefits a hypermedia application. Redundancy may provide more direct access points and thus improve navigation. This must be investigated further. RMM also offers no guidelines on user interface design.

We encountered two problems with indexes. First we had no way to specify the order index entries would display. For example, by default Ingres retrieves conference names in alphabetical order, where chronological order makes more sense—note the difference between the sketch in Figure 10a and the screen dump in Figure 2a. Thus we needed a way to think about and specify a rank ordering other than the DBMS's default. Second, RMM provided no way to specify intermediate or multi-level indexes. For example, we might wish to split authors into twenty-six separate indexes (A-Z) to shorten the number of entries for users, or we may wish to separate conferences by year. Furthermore, we need RMDM symbols for minimum slices, hybrid slices and user input. The latter would signify a user query to obtain a conditional search condition.

Because it models only systematic elements of applications, RMM provides no constructs or guidance for modeling and integrating elements that do not represent entity-relationship components. As with most Web applications, we had to hand-craft a central entry point to the system—LINKBase's home page, with links to ACM's homepage and to other non-automatable documentation such as an introduction, help information and acknowledgments. Such one-time elements constitute important "finishing touches" to applications. We would have appreciated some guidance on how to design them well and on how to integrate them into the systematic elements of our application. This includes both where to place access to them and which navigation tools to employ for this access.

We also found that RMM and this case study focus on structuring information for display, but not on how to maintain (add, delete and modify) information in the databases. As mentioned in step 7, we loaded the Ingres database tables through batch programs reading data files extracted (through FTP) from another bibliographic system. This data did not include author addresses and many of our other attributes. We have just started a project to collect (and maintain) data directly from input forms on the WWW. For example, authors would be able to update their own address information through the WWW. Maintaining author information is relatively straightforward, as all attributes belong to the same entity and thus to the same database table. Designing and processing the input forms for a new event item (including information about its parent event, sponsoring organization, publication and author) is more complex. Intuitively it seems that the same slice and user interface designs for displaying existing information could form the basis for designing input, deletion and modification screens. Because we designed the slice and user interface designs specifically to maintain context and reduce fragmentation, they may provide the proper presentation clustering for all user interaction with the system. In the end, however, we may decide it best to collect all information about an entity and its relationships on a single, long form. Maintenance thus presents interesting issues for future hypermedia design research.

6.2 Requirements Analysis: A Hypermedia Design Philosophy

RMM, as a design methodology, essentially begins once the requirements analysis phase is complete and the designers have determined the domain's entities and relationships. The exercise of designing LINKBase, as well as several other hypertext systems, prompted us to find a way to figure out which entities and relationships to include. In section 5.1 we noted that we started with the two entities, *Author* and *Conference_proceeding_article*, and ended up with the E-R design described in this case study.

While we are still refining our hypertext requirements methodology, we have come up with a preliminary set of guidelines which we have found useful in subsequent design projects. This has led us to a hypertext design *philosophy* or outlook on thinking about entities, attributes and relationships:

1. Analysts should do a standard requirements analysis as they normally would do for any application.
2. Analysts, together with the end-users, should consider each entity carefully. What are all possible pieces of information available for that entity? What are all possible things that different users (developers, content suppliers, end-users, those viewing on-line reports written by end-users) may wish to know about that entity? What are all things directly or indirectly related to that entity? Carefully thinking about these questions could yield additional attributes, additional incoming and outgoing relationships, and additional entities at the ends of these relationships. Bieber and Kacmar (1995) refer to this step as exercising hypertext's "philosophy of *maximum access*." Giving users freedom to access and explore as much information and metainformation as possible should help them comprehend the system better and have more confidence in its outputs.
3. Analysts, together with the end-users, should decide which of these new attributes, relationships and entities are both sensible and practical to include. It may be impractical to

collect data for some. Others may be too complex, redundant or indirect, or simply may not add enough value to include. In the end designers very well could be left with only a few additions, which still greatly enhance the application.

In sum, a hypertext design philosophy encourages designers to think of *things* in terms of their relationships, i.e., as if each thing were the center of a network of related information (including attributes and parameters) and other related things. In fact, this philosophy derives from the core concept of hypertext, which structures information in an associative network and gives users free access within that network.

6.3 Applying RMM to Legacy Systems

To date designers have used RMM for applications developed from scratch, and in which the entities and relationships developed for the screen layout match those underlying the application. Could we, then, apply RMM to retrofit the interaction with an organization's existing *legacy applications*—"large software systems that we don't know how to cope with but that are vital to our organization" (Bennett 1995)? In many legacy (and non-legacy) information systems, the screen layout does not reflect the system's internal computation or data structure. Yet their users would benefit from the more navigation, that RMM promotes.

We see two basic approaches to applying RMM to legacy systems: redesigning only the legacy system's interface, and rebuilding the entire system from scratch. Rebuilding entirely, while a very costly effort (Bennett 1995), gives an organization the opportunity to take advantage of years of experience with the domain to include missing features, as well as incorporating a hypermedia-based interface. In this case the RMM analysis must supplement the regular systems analysis and design process. Analysts would apply the regular design methodology to develop a requirements analysis and design the internal computational structure. Analysts would supplement the requirements analysis with a hypertext requirements methodology (see §6.2) to determine what entities, attributes and relationships to include. The analysts would then determine the system's presentation layout.

If choosing to replace only the legacy system's interface, the developer must fit RMM's hypermedia-based design with the existing legacy system's internal functionality. This involves four major steps. First the developer must determine how to pass information between the interface and the legacy system's functionality. If the legacy system was coded in a modular fashion that clearly separated the interface from the computational aspects (any other situation would be much more difficult, if not impossible), or if it has an API or batch mode interface, then the developer basically has to match each entity, entity attribute and relationship with the appropriate internal call (Bieber et al. 1995). Second, the developer should perform a hypertext requirements analysis to determine whether users would benefit from additional entities, attributes or relationships. The existing system effects a heavy constraint as any additional information must already exist somewhere in the system or be accessible from external sources. Most probably only additional relationships between existing entities will prove feasible. In this case users at least will benefit from more direct access to related information in displays. Third, an RMM analysis is to be performed to work out appropriate screen layouts and navigation design. Fourth, the user would employ the techniques determined in the first step to implement the RMM-based layout.

The label "legacy system" often reflects the desire of an organization to reengineer the system coupled with its frustration in the tremendous cost of doing so. When the concern primarily results from the system's interface as opposed to its internal functionality, retrofitting the interface with a hypertext-styled one may postpone or even alleviate the need to replace the legacy system in its entirety.

6.4 Summary of lessons learned

In addition to the lessons learned as part of the seven-step design process, we learned some general lessons which could be considered as general guidelines. As it turns out, some apply lessons already learnt by the information systems analysis and design community to hypermedia and WWW development.

- Analyze your application domain from a hypertext philosophy. Consider giving users direct access to every possible internal and external relationship, and to every possible related piece of information.
- RMM accommodates iterative design. Prototyping is essential to implementing interaction-intensive Web applications. Try alternate slicings, slice designs and access mechanisms. Chart out as many alternate navigation paths as possible between nodes (HTML pages).
- Maintain local context and minimize fragmentation. Plan all slices and node designs to minimize the amount of scrolling required to see the contents of a node. Ensure users can determine the kinds of information each node contains without scrolling. Cluster related information together. Provide titles or semantic labels when it is not absolutely clear what anchor values represent.
- Provide global context. Place all important items at the top (or bottom) of each Web document, possibly in a frame, to provide landmarks for orientation and navigation. Consider implementing a version of the RMD diagram as an overview map for navigation and orientation purposes.
- At or close to each external access point to your system (e.g., home page), provide access to major entry points ("grouping" access) for all major entities and relationships in your application domain.
- Do not feel obligated to implement every entity, attribute and relationship. To the extent possible, know all your users and the types of access they likely will find beneficial. This might help you decide how much information users would like to see, e.g., what to include in the minimal slice and head slice (thereby reducing the amount of navigation users require to get the information they desire). For diverse user groups, if necessary consider customized navigation options, and customized slice and node designs.
- Test each part of the system carefully, including each slice, structural link, association link, as well as any manual pages and *ad hoc* links (see §6.1.1).
- Use RMM in conjunction with other design and development methodologies (for user interface design, runtime behavior design, and construction and testing).

7. CONCLUSION

In this case study, we have demonstrated developing a Web-based information system using a systematic hypermedia design methodology. Although hypermedia itself symbolizes the free-form expression of ideas and relationships (Conklin 1995; Nielsen 1995; Selber 1995), the design and development of hypermedia applications require great discipline on the part of designers and developers. As *prima facie* evidence, one comes across many inconsistent and otherwise poorly implemented sites when browsing the World-Wide Web.

With the preponderance of WWW applications, it would be astonishing if not a single one were designed with a systematic methodology. Yet none of the research or practitioner literature describes a method for systematic hypermedia design. Perhaps the "hypertext" features are so intuitive that standard systems analysis and design techniques suffice, or the current set of applications are simple enough that standard design techniques suffice. Nevertheless, we believe that hypermedia interrelationships and functionality will prove easier and more robust to implement if done with a systematic methodology that explicitly revolves around these. We certainly have found this the case in our work, and we shall continue to improve upon hypermedia design methodologies in our research.

It is no secret that the information systems profession has benefited tremendously from the advances in systems analysis and design over the years. We hope that our work—and that of the rest of the hypermedia design community— can both convince the World-Wide Web community of the importance of systematic design, and provide them with a methodology for doing so.

References

Akscyn, R.M., McCracken, D.L., & Yoder, E.A. (1988). KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. Communications of the ACM, 31(7), 820-835.

Balasubramanian, V. & Turoff, M. (1995). A Systematic Approach to User Interface Design for Hypertext Systems. Proceedings of Twenty-Eighth Annual Hawaii International Conference on System Science (HICSS '95), Volume III, 241-250.

Bennett, K. (1995). Legacy Systems: Coping with Success. IEEE Software, January, 19-23.

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

Bieber, M. (1995). On Integrating Hypermedia into Decision Support and Other Information Systems. Decision Support Systems, 14, 251-267.

Bieber, M., & Kacmar, C.J. (1995). Designing Hypertext Support for Computational Applications. Communications of the ACM, 38(8), 99-107.

Bieber, M., & Kimbrough, S.O. (1992). On Generalizing the Concept of Hypertext. Management Information Systems Quarterly, 16(1), 77-93.

Botafogo, R., Rivlin, E., & Shneiderman, B. (1992). Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics. ACM Transactions on Information Systems, 10(2), 142-180.

Brown, P. (1990). Assessing the Quality of Hypertext Documents. In A. Rizk, N. Streitz & J. André, Eds. Hypertext: Concepts, Systems and Applications, Proceedings of European Conference on Hypertext (ECHT) '90, 1-12. Cambridge University Press.

Brown, J. S., & Newman, S. E. (1985). Issues in Cognitive and Social Ergonomics: From Our House to Bauhaus. Human-Computer Interaction, 1, 359-391.

Conklin, J. E. (1987). Hypertext: a Survey and Introduction. IEEE Computer 20(9), 17-41.

DeRose, S. (1989). Expanding the Notion of Links. Proceedings of Hypertext '89, 249-257.

Díaz, A., & Isakowitz, T. (1995). Computer-Aided Support for Hypermedia Design and Development. Proceedings of the International Workshop on Hypermedia Design, 3-15.

Elmasri, R., & Navathe, S. (1990). Fundamentals of Database Systems. Second Edition. Benjamin/Cummings Publishing Company,.

Garzotto, F., Mainetti, L., & Paolini, P. (1995). Hypermedia Design, Analysis and Evaluation Issues. Communications of the ACM, 38(8), 74-86.

Garzotto, F., Mainetti, L., & Paolini, P. (1996). Navigation in Hypermedia Applications: Modelling and Semantics. Forthcoming in Journal of Organizational Computing.

Garzotto, F., Paolini, P., & Schwabe, D. (1993). HDM - A Model-based Approach to Hypermedia Application Design. ACM Transactions on Information Systems, 11(1), 1-26.

Isakowitz, T. (1993). Hypermedia in Information Systems and Organizations: A Research Agenda. Proceedings of Twenty-Sixth Annual Hawaii International Conference on System Science (HICSS '93), Volume III, 361-369.

Isakowitz, T. and Kauffman, R., R. (1997). `Supporting Search for Reusable Software Objects. , IEEE Transactions on Software Engineering. (Forthcoming.)

Isakowitz, T., Stohr, E., & Balasubramanian, P. (1995). RMM: A Methodology for the Design of Structured Hypermedia Applications. Communications of the ACM, 38(8), 34-44.

Kahn, P. (1995). Visual Cues for Local and Global Coherence in the WWW. Communications of the ACM, 38(8), 67-69.

Kimbrough, S. O., Pritchett, C., Bieber, M., & Bhargava, H. (1990). The Coast Guard's KSS Project. Interfaces, 20(6), 5-16.

Lange, D. B. (1996). An Object-Oriented Design Approach for Developing Hypermedia Information Systems. Forthcoming in Journal of Organizational Computing.

Liu, C., Peek, J., Jones, R., Buus, B., & Nye, A. (1994). Managing Internet Information Services. California: O'Reilly & Associates.

Malcolm, K.C., S. E. Poltrock, S.E., & Schuler, D. (1991). Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise. Proceedings of Hypertext '91, 13-24.

Marshall, C., & Shipman, F. (1995). Spatial Hypertext: Designing for Change. Communications of the ACM, 38(8), 88-97.

Mao, J., Benbasat, I., & Dhaliwal, X. (1996). Enhancing Explanations in Knowledge-based Systems with Hypertext. Forthcoming in the Journal of Organizational Computing.

Minch, R. (1989-1990). Application and Research Areas for Hypertext in Decision Support Systems. Journal of Management Information Systems, 6(3), 119-138.

Minch, R., & Green, G. (1996). An Investigation of Hypermedia Support for Problem Decomposition. Forthcoming in the Journal of Organizational Computing.

Nanard, J., & Nanard, M. (1995). Hypertext Design Environments and the Hypertext Design Process. Communications of the ACM, 38(8), 49-56.

Nielsen, J., & Sano, D. (1994). SunWeb: User Interface Design for Sun Microsystems' Internal Web. Proceedings of the Second World-Wide Web Conference.

Nielsen, J. (1995). Multimedia and Hypertext: The Internet and Beyond. AP Professional.

Parunak, H.V.D. (1991). Toward Industrial Strength Hypermedia. In E. Berk & J. Devlin, Eds. Hypertext/Hypermedia Handbook, 381-395. New York: Intertext Publications/McGraw-Hill Publishing Co.

Rao, U., & Turoff, M. (1990). Hypertext Functionality: A Theoretical Framework. International Journal of Human-Computer Interaction, 2(4), 333-358.

Raskin, J. (1987). The Hype in Hypertext. Proceedings of Hypertext'87, 325-330.

Rettig, M. (1994). Prototyping for Tiny Fingers. Communications of the ACM, 37(4), 21-27.

Schloss, R. (1996). Novel Business Uses of Independently Created Hyperlinks. Proceedings of the 29th Annual Hawaii International Conference on System Sciences (HICSS '96).

Schwabe, D., & Rossi, G. (1995). Building Hypermedia Applications as Navigational Views of Information Models. Proceedings of the 28th Annual Hawaii International Conference on System Sciences (HICSS '95).

Schwabe, D., Rossi, G., & Barbosa, S.D.J. (1996). Systematic Hypermedia Application Design with OOHDM. Proceedings of Hypertext '96, 116-128.

Selber, S.A. (1995). Metaphorical Perspectives on Hypertext. IEEE Transactions on Professional Communication, 38(2), 59-67.

Thüring, M., Hannemann, J., & Haake, J. M. (1995). Designing for Comprehension: A Cognitive Approach to Hypermedia Development. Communications of the ACM, 38(8), 57-66.

Winograd, T. (1995). From Programming Environments to Environments for Designing. Communications of the ACM, 38(6), 65-74.

References on the World-Wide Web

CommerceNet. (1996). http://www.commerce.net/.

Johns Hopkins University Bioinformatics Web Server. (1996). http://www.gdb.org/.

WAIS, Inc. (1995). WAIS for UNIX, Chapter 1, Introduction. http://www.wais.com/company/Tech_chap1.html.

Information Dimensions, Inc. (1995). BASIS WEBserver™ Overview. http://www.oclc.org/oclc/idi/basisweb/8012web.htm.

Verity, Inc. (1995). Product Datasheet on Topic® WebSearcher. http://www.verity.com/datasheet.html.

NCSA. (1995). The Common Gateway Interface, National Center for Supercomputing Applications, University of Illinois-Urbana Champaign. http://hoohoo.ncsa.uiuc.edu/cgi/overview.html.

Ohio State University. (1996). The HCI Bibliography Project by Gary Perlman. http://www.cis.ohio-state.edu/~perlman/hcibib.html.

Department of Information Systems. (1996). New York University's Information Systems Department Handbook. http://is-2.stern.nyu.edu/isweb.