



PERGAMON

Information Systems 26 (2001) 295–320



www.elsevier.com/locate/infosys

## A case study in systematic hypermedia design

Venkatraman Balasubramanian<sup>a,\*</sup>, Michael Bieber<sup>b</sup>, Tomás Isakowitz<sup>c</sup>

<sup>a</sup>*E-Papyrus Inc., 1 Gramercy Road, Monmouth Junction, NJ 08852, USA*

<sup>b</sup>*Collaborative Hypermedia Research Laboratory, Information Systems Department, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA*

<sup>c</sup>*Research Department, Janney Montgomery Scott, 1801 Market St., Philadelphia, PA 19103, USA*

---

### Abstract

Hypermedia structuring and navigation requires design methodologies different from those developed for standard information systems. This case study 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 hypermedia-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, discuss human–computer interaction (HCI) aspects of RMM, and ground our work in the hypermedia design and HCI literature. Our experiences should encourage hypermedia and WWW developers to utilize systematic design techniques to build highly usable and useful WWW applications. © 2001 Elsevier Science Ltd. All rights reserved.

*Keywords:* Hypertext; Hypermedia; Design methodologies; Design guidelines; Entity-Relationship diagrams; Structure; Content; Navigation; User interface; Browsing semantics; Information access; Indexes; Guided tours; World Wide Web database gateways; LINKBase; Usability evaluation; Functionality evaluation

---

### 1. Introduction

Organizations are increasingly seeking to exploit hypermedia<sup>1</sup> to augment users' access to information. For example, we find hypermedia-based information kiosks in museums, airports and other public places. Hypermedia techniques help software developers find program segments for reuse [1]. Hypermedia structuring gives executives and analysts ready access to details and explanations

within decision support systems [2–10]. On the World Wide Web (WWW), organizations are exploring the potential of hypermedia access to present themselves to the public, and to sell their products and services [5]. Yet, while the field of hypermedia itself is maturing, methodologies for hypermedia design are beginning to emerge only now. To date, no published work has employed a systematic hypermedia design methodology for WWW development. Also, only relatively recently have there been reports about the need to incorporate human–computer interaction (HCI) and usability aspects into the WWW [11,12].

This article details our successful application of a hypermedia systems analysis and design

---

\*Corresponding author.

<sup>1</sup> While the term hypermedia nominally encompasses multiple media, we make no distinction between hypertext and hypermedia in this article.

methodology to a WWW application. We present a case study using the relationship management methodology (RMM) by Isakowitz et al. [13], propose extensions to RMM, summarize the “RMM design” lessons we learned in this process, describe HCI implications of RMM, and ground our experience in the hypermedia design and HCI 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 [14,15].

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 [16], and the possibility of errors and omissions [17,18]. Therefore, a systematic design approach with automatic page generation constitutes the only reasonable option in many cases.

Isakowitz et al. [13], 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 [19] and Johns Hopkins University’s [20] public domain bioinformatics databases), 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 cannot learn from them.

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

## 2. The relationship management methodology (RMM)

The 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 [13].

### 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) conversion protocol 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 design: how to populate the application with data and how to provide interaction behavior, and (7) construction and testing: actual development of programs and testing.

RMM was conceived to be flexible by supporting rapid feedback loops as prescribed in [21]. This is embodied in its software design tool, RMCASE [22]. Although not explicit in the methodology,

HCI aspects are an integral part of RMM and we will present these in the discussion of each of the seven steps.

2.2. The RMDM data model

The RMDM is the cornerstone of the RMM methodology. Fig. 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 RMD diagram (see Fig. 9). The RMDM model is based on the entity-relationship model [23], and on HDM [24] and HDM2 [25].

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. Splitting an entity into slices also reduces clutter of the information display. Only the most essential information is presented upfront with details being available as links.

RMDM specifies navigation via the six access primitives at the bottom of Fig. 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

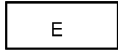


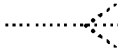
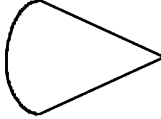


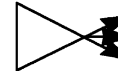

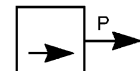

<b>E-R Domain Primitives</b>	Entity	
	Attribute	
	One-One Associative Relationship	
	One-Msny Associative Relationship	
<b>RMD Domain Primitives</b>	Slice	
<b>Access Primitives</b>	Uni-directional Link	
	Bi-directional Link	
	Grouping	
	Conditional Index	
	Conditional Guided Tour	
	Conditional Indexed Guided Tour	

Fig. 1. The elements of the RMM data model (RMDM).

tours. Logical conditions qualify these access structures. For example, attaching the condition “type=‘panel’” to an index of an Event entity denotes an index to panels from that event. 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 identify entity instances uniquely. 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 these new kinds of slices in Section 5.4.

## 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 et al.’s HDM data model [24] and its successor HDM2 [25] describe 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 [3] and Schwabe et al. [26], 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 are already familiar. In fact, as we note in Section 5.10 designers using RMM especially have lauded this familiarity.

At the other extreme, Marshall and Shipman [8] 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 RMD diagram (Section 5.5), as does RMCASE [22]. The interaction between the structured RMM methodology and spatial hypertexts remains to be explored.

Our experience follows Nanard et al.’s 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 [21]. Employing RMM certainly had both aspects. Nanard et al. 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 direct the reader to our prior research in systematic hypermedia user interface design [27] and in automatically generating hypermedia based on an application's internal structure [28,29]. These do not have the breadth of RMM. Lastly we note that researchers are beginning to focus on the need to incorporate design and usability aspects into WWW applications from an HCI point of view [11,30].

#### 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.

The following scenario illustrates LINKBase. A user interested in information about hypermedia reaches ACM SIGLINK's LINKBase. 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 Fig. 2a, which contains an index of hypermedia related conferences. Upon selecting "ECHT '94" (European Conference on Hypermedia Technology 1994), the system generates the WWW page shown in Fig. 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 Fig. 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 Fig. 3b. From here the user can select author names and navigate to details about their affiliation and contact information.

#### 5. LINKBase case study

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 content and access mechanisms, 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.

This section describes LINKBase's design and development using RMM. We will present the relevance of HCI principles with respect to the seven design stages, where applicable.

##### 5.1. Requirements analysis

We, the designers, played the role of users (both content providers and readers) while discussing the requirements for the new LINKBase. The second author who implemented the original LINKBase knew its deficiencies: primarily, it lacked a lot of content relevant to hypermedia researchers. Access and update mechanisms were also very limited. Although usability aspects were not of primary

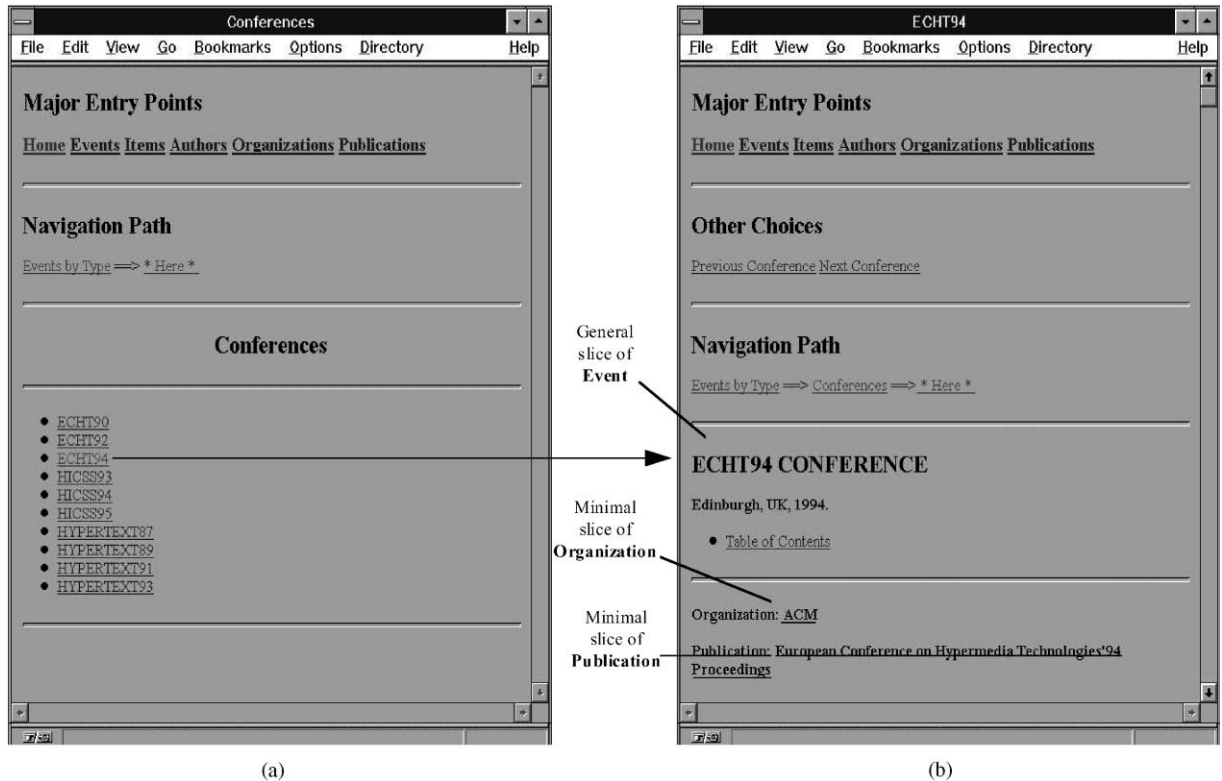


Fig. 2. Screenshots of a prototype LINKBase implementation showing navigation between dynamically generated Web pages. Screenshot (a) lists hypermedia research conferences; (b) results from selecting “ECHT’94,” and provides the conference’s date and location, and access to its table of contents.

importance while re-designing the system, as we discuss in Section 5.11, the discipline encouraged by RMM results in a system with a high-degree of usability and utility. The second author along with the first, and two students taking a course in hypermedia discussed the functionality and content required for a comprehensive bibliographic system on hypermedia. This approach is similar to other design approaches reported by Salomon [31], that is, the designer taking the role of the user. Based on discussions amongst ourselves, we arrived at the following new requirements for the system:

- Improve the information content of LINKBase by incorporating missing information. The original LINKBase had a list of conferences, list of papers and their abstracts and some information on authors. Also, all this informa-
- tion was restricted to events organized by the ACM. A lot of hypermedia related information was available in the form of workshop reports, other conferences, books, technical reports, etc. In addition to these other sources, we also wanted to provide complete author information (such as affiliation, contact address, e-mail, etc.), more information about the event (location, dates), event managers, sponsoring organizations (ACM, IEEE, universities, etc.), and resulting publications. Being researchers ourselves, we felt that these would be additional pieces of information in which other researchers would be interested.
- Provide easy mechanisms for content providers to update information about themselves and the organizations they represent.
- Provide an easy-to-use and intuitive interface using the World Wide Web as the medium so

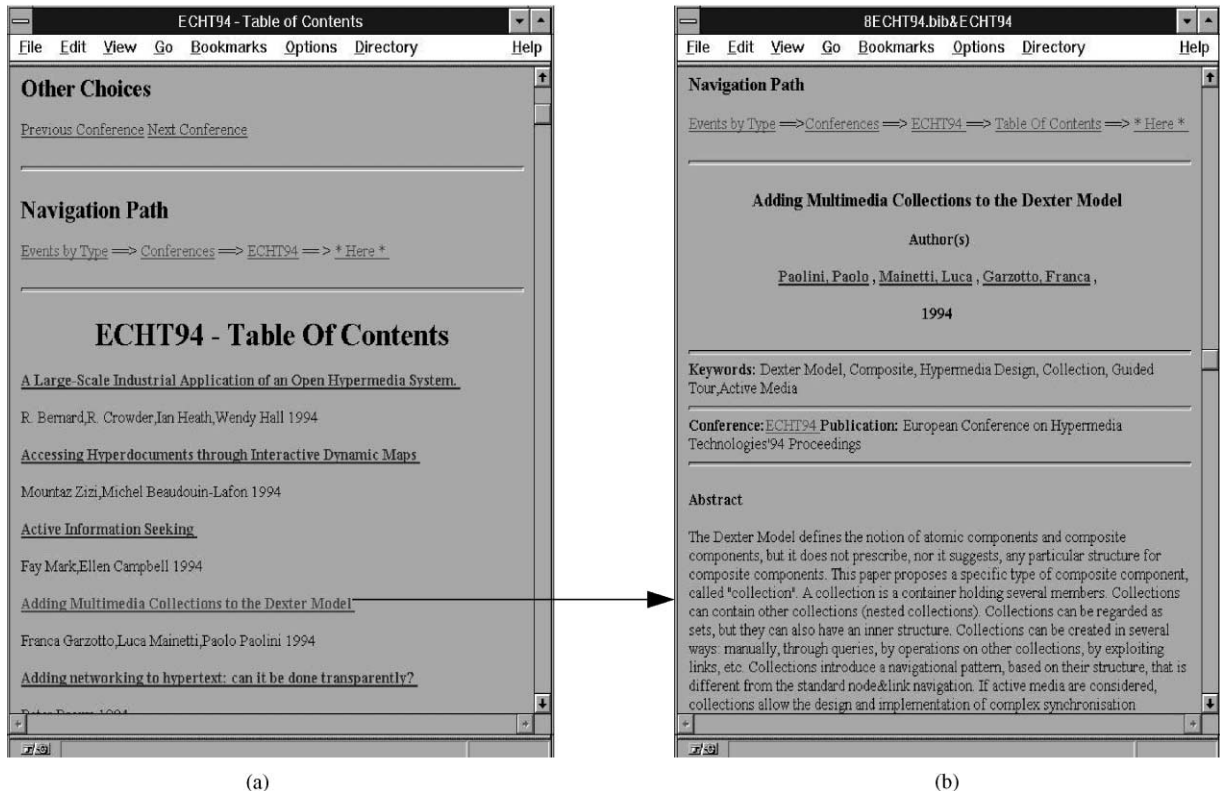


Fig. 3. Screenshot (a) shows the table of contents for ECHT '94 selected from Fig. 2b. This lists all papers, posters, demonstrations and videos presented at the conference. When the user selects the paper "Adding multimedia collections to the dexter model," the system generates screenshot (b). It shows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

that the information is easily accessible by researchers all over the world.

- Develop the system in a systematic manner so that it is easy to maintain and update.
- Generate HTML documents dynamically from an up-to-date source so that HTML documents need not be created and maintained manually.

Hence, the primary purpose of re-implementing LINKBase was to provide additional information to researchers interested in the field of hypermedia, increased functionality in terms of various ways to access the information, a simple-to-use user interface, and an easy update mechanism.

### 5.2. Why use RMM to re-implement 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<sup>TM</sup> [32] or BASIS WEBserver<sup>TM</sup> [33] or Topic<sup>®</sup> WebSearcher [34] successfully support good indexing and searching facilities on variable length text strings, they neither support the design process nor the management of relationships.

LINKBase satisfied the criteria for an RMM-based design specified in [13]. The domain's structure contains no ad hoc relationships (except for our manually added introduction and acknowledgments, etc.—see Section 6.1), 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 [6,35], 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 Section 6.1) instead of letting (numerous) people update HTML pages at will and potentially degrading LINKBase's consistent layout and integrity.

### 5.3. 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 system analysts with experience in developing applications employing relational databases. It results in an E-R diagram for the entire application.

#### 5.3.1. 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 in which it was presented.

Based on these requirements, we identified five entities *Organization*, *Event*, *Event\_item*, *Person*, and *Publication* and eight associative relationships<sup>2</sup> Fig. 4 shows the 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 Fig. 4 follow a similar pattern.

#### 5.3.2. Lessons learned

E-R Design is nothing but data modeling in the traditional sense with a slight modification. In the case of E-R design for hypermedia applications, there may be redundancy in the way entities are related to each other in order to facilitate navigation. Although we can indirectly find out where an *Event\_item* can be found in a *Publication* through the associated event entity, we added an additional relationship between *Event\_item* and *Publication* entities so that we facilitate direct navigational access. Therefore, even a purely data-oriented phase such as E-R Design takes into account the navigational aspect which has been of primary importance for both HCI and WWW researchers. E-R design focuses on what information must be presented to the reader while the following phase, Slice Design, focuses on how much of it must be presented at a given time.

### 5.4. 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. Slice design is similar to chunking information appropriately for presentation purposes. It is important that these slices are not only self-contained but also form cross-references to other slices or chunks [36]. Each slice by itself should be coherent on its own. When slices are combined

<sup>2</sup>Entity and relationship names are in italics where they are used in the context of LINKBase. Also note that relationships are shown in both directions.



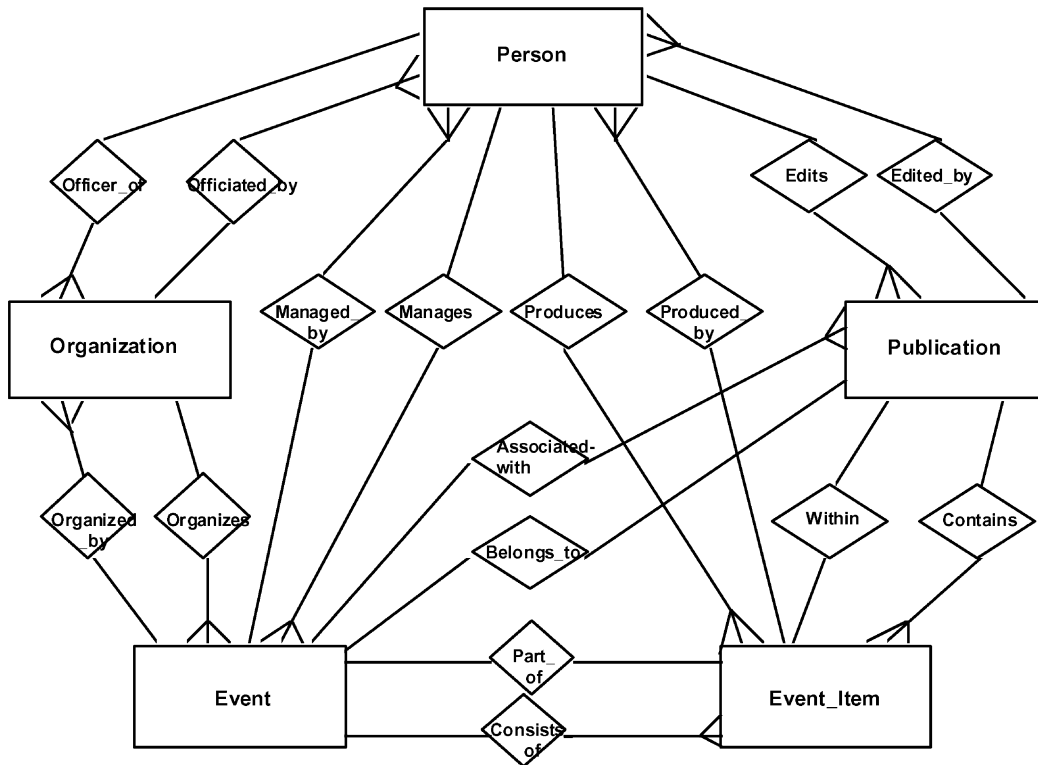


Fig. 4. E-R Diagram for LINKBase application.

together, the combined view should also be coherent. Presenting fewer slices on the display will reduce clutter and scrolling, and overcome short term memory limitations [37]. This potentially can increase comprehension of the displayed information. 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- or bi-directional. Appropriate labeling of links 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.

5.4.1. LINKBase example

Since they are unique contributions of this case study, we describe how minimal and hybrid slices were used. Fig. 5 shows the slice diagram for the Event entity. Its minimal slice contains two attributes: EventName and EventType, for exam-

ple, “ECHT’94” and “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.

Displaying information about Event\_Item illustrates the use of hybrid slices. Fig. 6 shows an instance of an event item as seen on a browser. Its schematic screen or node design, as shown in Fig. 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(Name), refers to the minimal slice of the Person entity related to the Event\_Item via the Produced\_by relationship from Fig. 4’s E-R diagram. The name attribute (in this case, PersonLastName and PersonFirstName)

Table 1  
Entity and relationship attributes in LINKBase's domain

Entity/relationship	Attributes
Organization	OrgName, OrgType (University, SIG, professional organization, etc.), Address, Phone, Fax, E-mail, 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

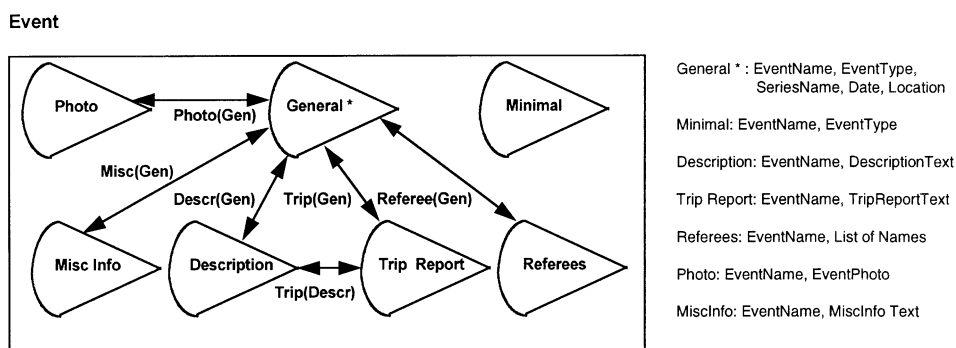


Fig. 5. Slice diagram for the Event entity showing its slices and their attributes. The general slice serves as the head slice. The minimal slice contains the minimum set of context information necessary to identify this entity when embedding it within hybrid slices of other entities.

forms the anchor to a link to the Person entity. More generally, the format is: “(relation) entity → slice name (attribute anchor).”

A hybrid slice is conceptualized as a composition of slices (pure or hybrid) from other entities,

that is anchored in one specific entity. Fig. 8 shows the design of the hybrid slice we have been discussing. It is anchored in Event\_item. Fig. 8 illustrates how its design combines slices from the Person, Event\_item, Event and Publication entities.

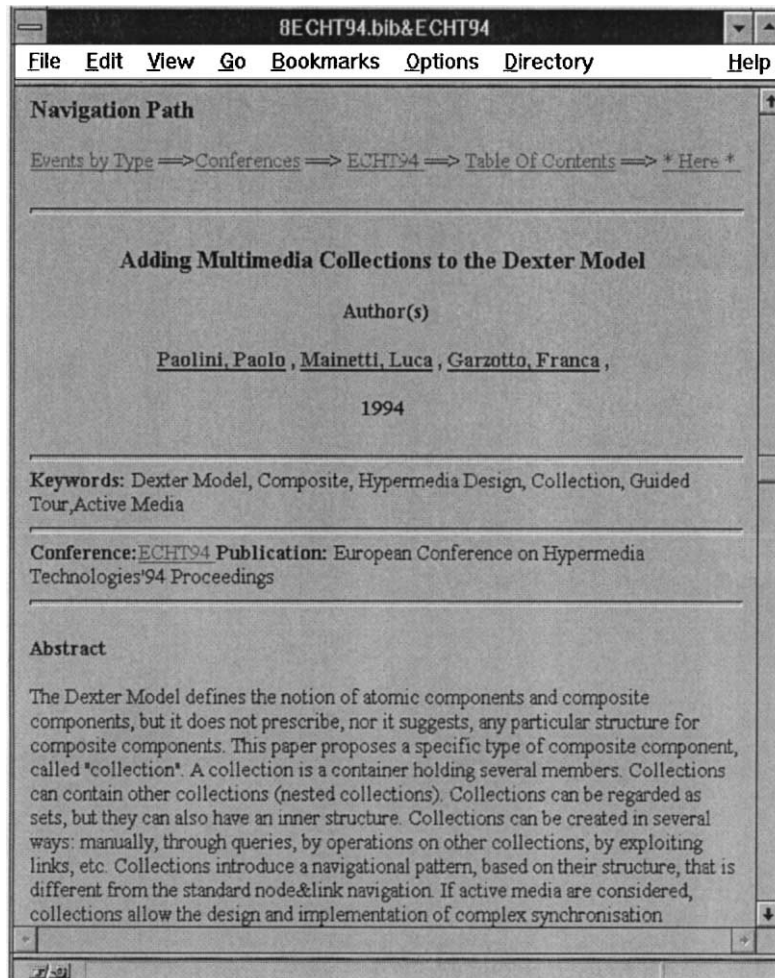


Fig. 6. An instance of a hybrid slice as viewed in a browser.

#### 5.4.2. Lessons learned

Slices and hybrid slices determine how much information must be displayed. This simplifies the user interface design stage discussed later. During slice design the designer begins to refine and prune the application's design. For example, one might decide to drop some attributes included in the E-R diagram. For example, we dropped the biography attribute from Person and subject classification from Event\_Item, as both would 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 [36] and reducing local coherence [30,38]. This tradeoff is the subject of a long-standing philosophical and user interface-oriented debate in the hypermedia community [39–41]. 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 (and without

overwhelming 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.

5.5. Navigation design

The third step, navigation design, involves identifying the paths that enable hypermedia 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. Navigation design is related to user interface design. While the former focuses on providing easy access mechanisms to various entities and their relationships, the latter focuses on presentation aspects of the information. Navigation must occur within an entity (via structural links) and between entities (via associative links). 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

- Event\_item->General**
- (Produced By) Person->Minimal(Name)**
- (Part\_of) Event->Minimal(Name)**
- (Within) Publication->Minimal(PublicationName)**
- (Content)**
- (Reference Information)**
- (Miscellaneous Information)**

Fig. 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 (attribute anchor).” The underlined words represent the attributes that serve as anchors to links to other slices.

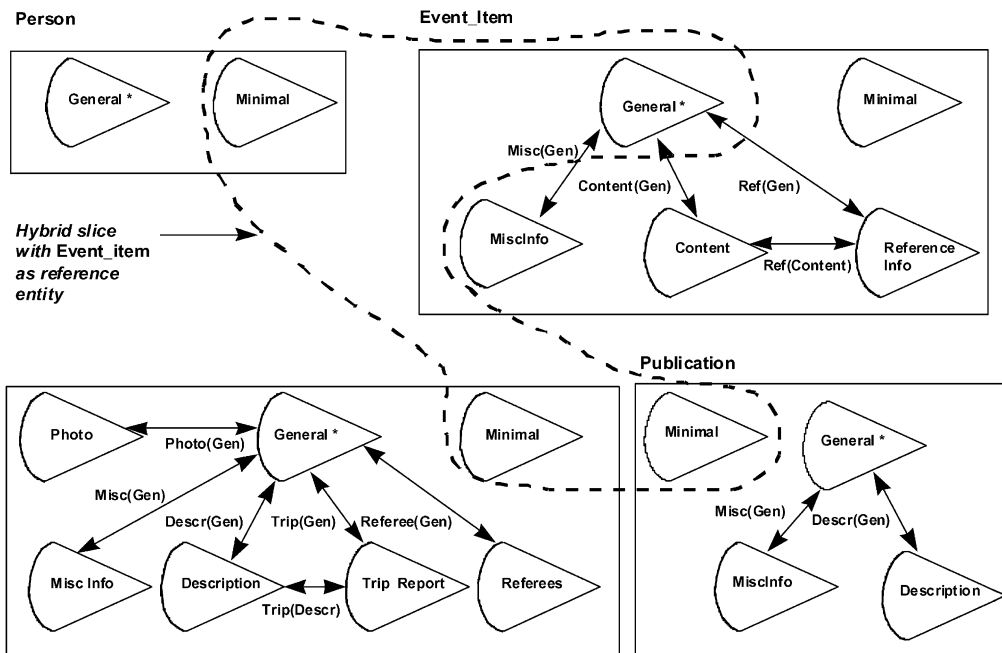


Fig. 8. The general slice of Event\_item and the minimum slices of Event, Person, and Publication have been combined together (covered by the dashed lines) to form Fig. 6 and Fig. 7’s hybrid slice representing an Event\_item.

links—then the RMDM unidirectional access mechanism suffices. Each of the access paths are represented by one of Fig. 1’s RMDM access structures. This step produces an RMD diagram.

5.5.1. LINKBase example

Fig. 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,” and then all

papers presented there using the 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.

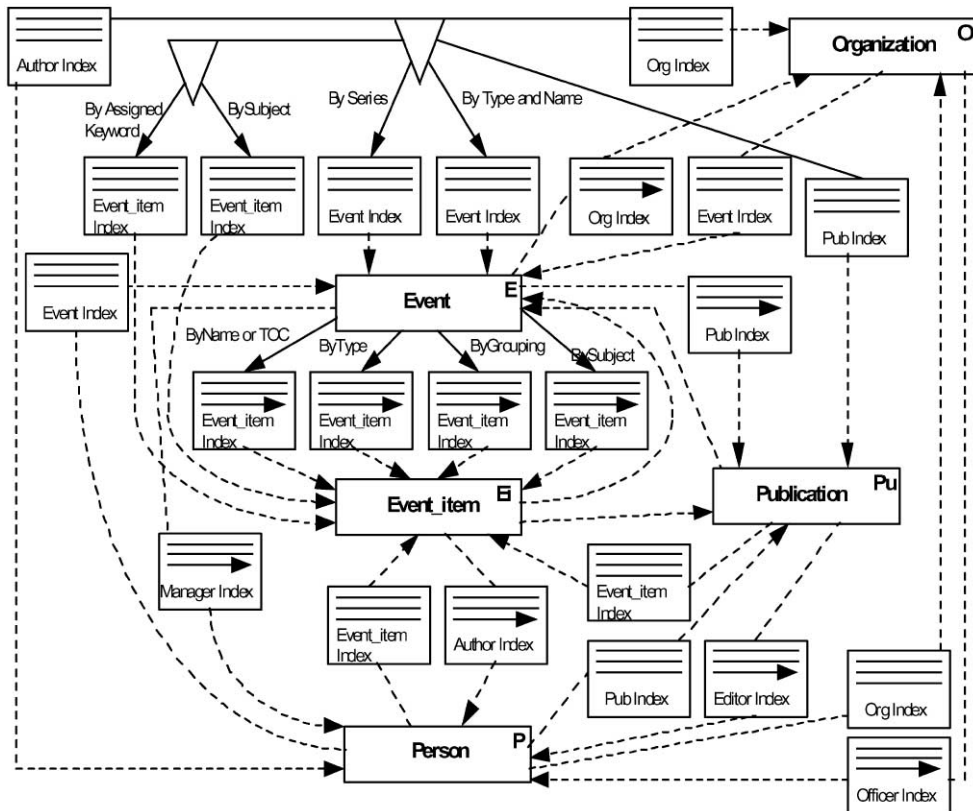


Fig. 9. RMD Diagram for LINKBase.

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 Person entity, restricting major access to only authors.

#### 5.5.2. 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.

During navigation design, we also decided on the qualifier attributes for accessing information based on a reader’s queries or needs. For example, we provided facilities for the reader to access an Event\_item index as a plain old vanilla table of contents, by type of the Event\_item (for example, ‘paper’), by keywords, etc. We also included the access mechanism where one could access an Event by its series name (for example, ‘ACM’).

#### 5.6. Conversion protocol design

Isakowitz et al. [13] 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 programs 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.

##### 5.6.1. 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).

##### 5.6.2. 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 the departmental information system illustrated by Isakowitz et al. [13] and implemented in [42]. 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.

### 5.7. Step 5: User interface design

The fifth step, user interface design, involves designing presentations for every RMD diagram object. This involves how to present the following: the information content of slices and hybrid slices identified in the second step; access structures to linked information from the third step; location of orientation and navigation components; and layout of link anchors and nodes.

Just like any other application, one could employ low-fidelity prototyping using paper and pencil strategies to prototype the interface [43] or employ high-fidelity prototyping with computerized tools [44]. The information identified in the form of slices and hybrid slices must be prototyped. In addition, the access mechanisms and groupings identified during navigation design should be included.

This phase, along with navigation design, is similar to constructing an interaction state diagram but less formal. Representations of different display objects along with navigation mechanisms between them can be represented in the form of user interface design and navigation charts through paper and pencil prototypes [45]. This provides a high-level overview of how various pieces of information can be accessed but does not necessarily include a state diagram. By default, the head slice of an entity is always displayed as part of a traversal from one entity to another. It acts as an index to other slices of the same entity. As we can see, both navigation design and user interface design are closely related; and we frequently interacted between them.

#### 5.7.1. LINKBase example

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

In order to understand the flow of HTML pages and various navigation pathways better, we con-

structed 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 an HTML page in our application. Thus, we adopted both low- (paper) and high-fidelity (working) prototyping techniques.

We adopted many user interface design guidelines suggested by hypermedia researchers. To maintain a sense of orientation and facilitate navigation, we followed the guidelines for local and global context by Kahn [38] and Thüring et al. [30]. 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 employed a stable screen layout, as prescribed by Thüring et al.'s eighth hypermedia design principle [30]. We placed global links under the header "Major Entry Points," which contains all the major groupings from Fig. 9's RMD diagram. Major entry points provide the reader with landmarks [46] or strategic choices [27], which are always visible. To minimize clutter and scrolling, and hence user interface adjustment [30], we placed all structural link anchors to other slices under the heading "Other Choices." Grouping of choices or menu items have been discussed widely by HCI researchers [37]. 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 [42] 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 [47]. Instead, we chose simple textual link anchors.

#### 5.7.2. Lessons learned

Because Web browsers at the time did not offer frames and we chose to use a single window per

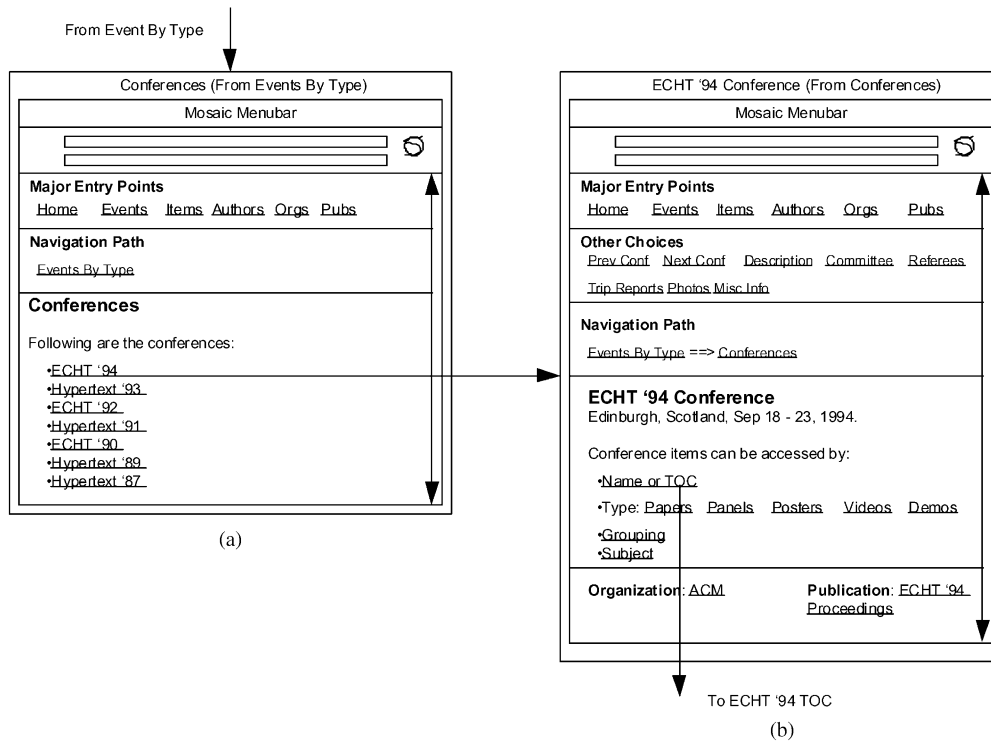


Fig. 10. User interface design and navigation pathways: Low-fidelity sketches of our windows. The window with a list of conferences in (a) is generated from the “Homepage” upon selecting “Events By Type” and then “Conferences.” It prototypes the general layout for an index. Clicking on a conference name 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 associative links to indexes of this conference’s Event\_items.

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 entirely 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.

### 5.8. Step 6: Runtime behavior design

In the sixth step, runtime behavior design, one designs the programs that will control how the 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



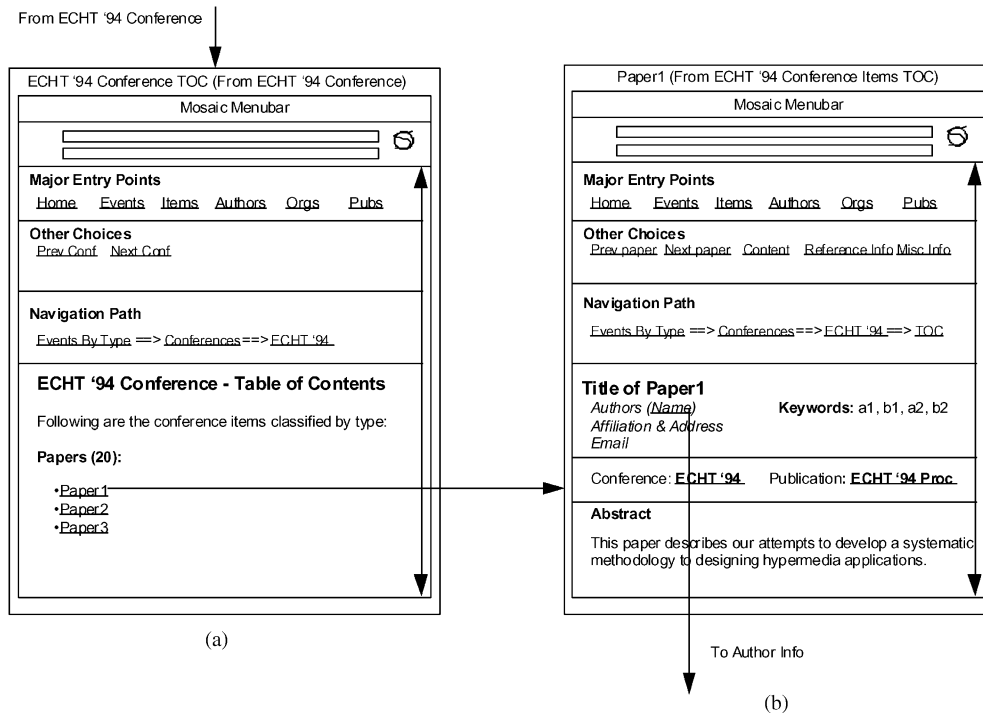


Fig. 11. User interface design and navigation pathways, continued. The window in (a) prototypes the general layout of an indexed guided tour's leading index page. It is generated by selecting "Table of Contents" in Fig. 10b. It shows the Event.items grouped by type such as papers, posters, etc. The window in (b) portrays a page along with the indexed guided tour (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 hybrid slice design for an Event.item from Fig. 7. The navigation path in each window reflects the reader's traversals.

designing the algorithms and implementation mechanisms for hypermedia navigation such as browsing (link traversal), history tracking, indexed navigation, guided tours and backtracking (such as our navigation path feature). The designer should identify those parts of the information content that need to be generated dynamically, which 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.

### 5.8.1. 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 [4].

To preserve both local context and current position (design principles three and six in [30]) we added "navigation path," which the system dynamically generates based on the user's link traversals. For example, the table of contents window in Fig. 8a has the navigation path "Events By Type  $\Rightarrow$  Conferences  $\Rightarrow$  ECHT '94." The current implementation appends " $\rightarrow$  \*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 [48], 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 currently available popular browsers.

### 5.8.2. Lessons learned

Runtime behavior design allowed us to take the output of user interface design (a series of HTML documents) and implement browsing semantics in an incremental way and thereby observe runtime behavior. We prototyped a few runtime behaviors and iterated through a few rounds of construction and testing (step 7) so that we could observe the dynamic behavior of our C programs and interfaces to CGI. Most of our lessons learned were technical. Since this was the first time we interacted with a database through CGI, we learned 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 are different. We especially relied on Web resources concerning the CGI [49] and Liu et al. [43]. Gateways to RDBMS backends can now be implemented through application programming interfaces (API) provided by popular Web servers.

### 5.9. 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 testing each link to ensure that we generate the proper underlying parameters.

#### 5.9.1. 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 hypermedia field from the HCI bibliography project files [50]. 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.

Fig. 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 Web browser detects when the reader selects a link and passes it to the Web server. For example, consider Figs. 2 and 10. 2(a) and 10(a) show the list of hypermedia conferences, that is events selected by type = ‘conference’. Assume the user requests a list of all papers (EventItems of type = ‘paper’) associated with ECHT’94. The Web server invokes a csh script, which then invokes a C program, passing the arguments “ECHT’94” and “paper” through the Web server environment variable QUERY\_STRING. The program treats these arguments as command-line arguments, which it then passes as variables to embedded SQL (ESQL)

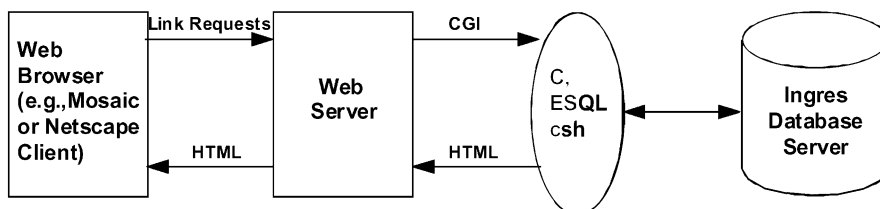


Fig. 12. LINKBase’s client/server architecture.

statements. These SQL queries then execute over the database. The Ingres server returns results (rows of paper titles and author names). The C program processes the results, adds appropriate HTML tags and converts them to HTML “on the fly.” Those parts of the result’s text which need to be identified as link markers will be replaced by an anchor containing a URL for its appropriate C program and arguments to be passed to it. The Web server is then informed that the stream coming through the standard output is an HTML document, that is, the “content-type” is “text/html.” The Web server forwards the HTML document across the network to the client Web browser for display. Thus, links become queries. In this way, LINKBase dynamically generates each HTML document (with a few exceptions covered in Section 6.1) 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.

#### 5.9.2. *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 entities, relationships, navigational mechanisms, and views. This experience refined our own understanding and belief in RMM’s iterative and systematic approach to LINKBase’s development.

#### 5.10. *RMM usability*

RMM made it easy for both the designers and developers to build a highly usable and functional bibliographic hypermedia system. In preparation for this paper, developers of LINKBase were interviewed (informally) about RMM’s usefulness and usability.

LINKBase’s primary developer who is the first author of this paper said, “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 [26]) 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 important 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 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 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 Section 6.1, the first implementor was frustrated that RMM provides so little help on implementing the design.

#### 5.11. *Usability evaluation of LINKBase: HCI implications of RMM*

While we did not focus explicitly on improving the usability aspects of the original LINKBase, employing RMM resulted naturally in increased usability in addition to the better content and more functionality which was our original

requirement. However, this does not imply content is more important than usability [12].

Garzotto et al. describe five dimensions for systematic design: an application's content, structure, presentation, dynamics and user interaction [16]. They then provide several heuristics for evaluating hypermedia applications in these terms, including richness, consistency and self-evidence. RMM's approach leads developers to design all five dimensions, and, we believe, promotes design habits that would score high evaluation ratings. Table 2 shows the stages at which RMM incorporates each of the five design dimensions. Conversion protocol design and construction and testing are more implementation steps rather than design steps. As designers, we conducted informal usability evaluation by employing heuristic techniques as part of the testing stage. Once again, this was the case of designers becoming user interface specialists. The new LINKBase satisfied some of the system acceptability parameters required for hypermedia [51] such as usability (both in terms of content and structure), utility or functionality (in terms of access mechanisms and multiple pathways), and an easy-to-use browser-based interface with well-designed orientation and navigation elements. We could not carry out extensive usability testing with real users since we are currently in the process of migrating LINKBase to a different RDBMS as the University decided to phase out Ingres.

Although not explicit in the methodology, six out of the seven RMM stages incorporate HCI related aspects. Based on our discussions, we see that E-R design includes identifying relationships

which do not occur naturally so as to facilitate navigation. It focuses on what information must be presented. Slice design focuses on chunking information appropriately to reduce clutter and increase comprehension. It focuses on how much information must be presented. Navigation design is interested in providing various access mechanisms or navigation pathways to retrieve information. It focuses on how to access various pieces of related information. User interface design includes presentation of content, structure, orientation and navigation elements. It focuses on how to present information. Runtime behavior design includes identifying browsing semantics and prototyping interaction states. It focuses on how to model browsing semantics and interaction behaviors. Construction and testing include usability evaluation in addition to testing for functionality. They focus on how to build and test the system. Thus, RMM helps build information systems that satisfy aspects of both functionality and usability [52]. RMM results naturally in hypermedia systems with increased usability and utility due to the discipline that it inculcates in the designer.

## 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 retro-fitting existing legacy applications with a hypermedia interface (Section 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.

Table 2  
RMM's seven stages compared to five evaluation dimensions of a hypermedia application [13]

Design dimensions [24]	RMM stages [13]
Content	Entity-relationship design
Structure	Slice design
Presentation	Navigation design and user interface design
Dynamics	Runtime behavior design
Interaction	Navigation design and runtime behavior design

While Isakowitz et al. lay the foundation for the seven-step RMM, they provide details only for the first three steps [13]. This article contributes by expanding on the remaining four. During slice design we added two intermediate steps of identifying minimal slices and hybrid slices. We also contributed the concept of making the logical data model physical during conversion protocol design. While HCI aspects have been implicit in RMM, we have brought out the relevance of HCI to RMM.

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 Section 6.1 by considering RMM's strengths and limitations, and by proposing some extensions. In Section 6.2 we expand on performing a hypermedia requirements analysis, which should precede any RMM analysis. In Section 6.3 we consider applying RMM to an organization's existing legacy systems. We close in Section 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 hypermedia philosophy of considering all possible (internal and external) relationships and maximum user access (see Section 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) [53] and system opacity (mismatch between the implementation model and the interface) [54].

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, although we augmented our design with guidelines by [38,46,37,30].

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 Fig. 10a and the screen dump in Fig. 2a. Thus we need 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 acknowledgements. 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 is 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 hypermedia systems, prompted us to find a way to figure out which entities and relationships to include. In Section 5, 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 hypermedia requirements methodology, we have identified a preliminary set of guidelines which we have found useful in subsequent design projects. This has led us to a hypermedia 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? Careful 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 refer to this step as exercising hypermedia's "philosophy of maximum access" [55]. Giving users freedom to access and explore as much information and meta-information as possible should help them comprehend the system better and have more confidence in its outputs. As designers, we should plan for providing maximum access to information, although practical considerations may dictate that such access may not be provided all at once. It can be provided in an incremental and iterative manner.
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.

Bieber formalizes points 2 and 3 as the relationship-navigation analysis, identifying several general relationship types found in most applications [56].

In summary, a hypermedia design philosophy encourages designers to think of objects in terms of their relationships, i.e., as if each object were the center of a network of related information (including attributes and parameters) and other related objects. In fact, this philosophy derives from the core concept of hypermedia, 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" [57]? 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 better navigation that is offered by RMM.

We see two basic approaches to applying RMM to legacy systems: re-designing only the legacy system's interface, and rebuilding the entire system from scratch. Rebuilding entirely, while a very costly effort [57], 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 hypermedia requirements methodology (see Section 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 [55]. Second, the developer should perform a hypermedia 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 re-engineer 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, retro-fitting the interface with a hypermedia-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 those learned as part of the seven-step design process, we learned lessons which could be considered as general guidelines. As it turns out, some apply lessons the information systems analysis and design community has already learned to hypermedia and WWW development:

- Analyze the application domain from a hypermedia 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 slice designs, hybrid slices, 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 hybrid slices 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 the users and the types of access they likely will find beneficial. This might help 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, associative link, as well as any manual pages and ad hoc links (see Section 6.1).
- 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. We have also discussed how RMM promotes usability and utility aspects of WWW applications due to the discipline that it encourages during the design process. Although hypermedia itself symbolizes the free-form expression of ideas and relationships [40,46,30], 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 hypermedia 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 researchers in the hypermedia design and HCI communities—can convince the World Wide Web community about the importance of systematic design and hypermedia usability, and provide them with a methodology for doing so.

## Acknowledgements

We would like to thank Bang-Min Ma, Yu Cheng and 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 New Jersey Center for Multimedia Research, by the National Center for Transportation and Industrial Productivity at NJIT under Grant # 990905, the New Jersey Department of Transportation, grants from the Sloan Foundation, the AT&T Foundation and the United Parcel Service, and the NASA JOVE faculty fellowship program.

## References

- [1] T. Isakowitz, R. Kauffman, Supporting search for reusable software objects, *IEEE Trans. Software Eng.* (1997) forthcoming.
- [2] S.O. Kimbrough, C. Pritchett, M. Bieber, H. Bhargava, The coast guards KSS project, *Interfaces* 20 (6) (1990) 5–16.
- [3] D.B. Lange, An object-oriented design approach for developing hypermedia information systems, *J. Organization. Comput. Electron. Commerce* 6 (3) (1996) 269–293.
- [4] C. Liu, J. Peek, R. Jones, B. Buus, A. Nye, *Managing Internet Information Services*, O'Reilly & Associates, CA, 1994.
- [5] G.L. Lohse, P. Spiller, Electronic shopping, *Commun. ACM* 41 (7) (1998) 81–87.
- [6] K.C. Malcolm, S.E. Poltrock, D. Schuler, Industrial strength hypermedia: requirements for a large engineering enterprise. *Hypertext '91 Proceedings*, ACM Press (1991) 13–24.
- [7] J. Mao, I. Benbasat, J.S. Dhaliwal, Enhancing explanations in knowledge-based systems with hypertext, *J. Organization. Comput. Electron. Commerce* 6 (3) (1996) 239–268.
- [8] C. Marshall, F. Shipman, Spatial hypertext: designing for change, *Commun. ACM* 38 (8) (1995) 88–97.
- [9] R.P. Minch, Application and research areas for hypertext in decision support systems, *J. Mgmt. Inform. Systems* 6 (3) (1989) 119–138.
- [10] R.P. Minch, G.I. Green, An investigation of hypermedia support for problem decomposition, *J. Organization. Comput. Electron. Commerce* 6 (3) (1996) 295–321.
- [11] K. Instone, HCI and the Web, *CHI '96 Trip Report*, *SIGCHI Bulletin* 28 (4) (1996) 42–45.
- [12] S.B. Shum, The missing link: hypermedia usability research and the web chi '96 trip report on the british hci group symposium, *SIGCHI Bull.* 28 (4) (1996) 68–75.
- [13] T. Isakowitz, E. Stohr, P. Balasubramanian, RMM: a methodology for the design of structured hypermedia applications, *Commun. ACM* 38 (8) (1995) 34–44.
- [14] M. Bieber, F. Vitali, Toward support for hypermedia on the World Wide Web, *IEEE Comput.* 30 (1) (1997) 62–70.
- [15] M. Bieber, F. Vitali, H. Ashman, V. Balasubramanian, H. Oinas-Kukkonen, Fourth generation hypermedia: some missing links for the World Wide Web. *Int. J. Human Comput. Studies* 47, 31–65.
- [16] F. Garzotto, L. Mainetti, P. Paolini, Hypermedia design analysis and evaluation issues, *Commun. ACM* 38 (8) (1995) 74–86.
- [17] R. Botafogo, E. Rivlin, B. Shneiderman, Structural analysis of hypertexts: identifying hierarchies and useful metrics, *ACM Trans. Inform. Systems* 10 (2) (1992) 142–180.
- [18] P. Brown, Assessing the quality of hypertext documents, in: A. Rizk, N. Streitz, J. André, (Eds.), *Hypertext: Concepts, Systems and Applications*, Proceedings of the European Conference on Hypertext (ECHT) '90, 1–12. Cambridge University Press, 1990.
- [19] CommerceNet, <http://www.commerce.net/>, (1996).
- [20] Johns Hopkins University Bioinformatics Web Server, 1996, <http://www.gdb.org/>.
- [21] J. Nanard, M. Nanard, Hypertext design environments and the hypertext design process, *Commun. ACM* 38 (8) (1995) 49–56.
- [22] A. Díaz, T. Isakowitz, Computer-Aided Support for Hypermedia Design and Development. Proceedings of the International Workshop on Hypermedia Design 1995, LIRMM, 3-15, 1995.
- [23] R. Elmasri, S. Navathe, *Fundamentals of Database Systems*, 2nd Edition, Benjamin/Cummings Publishing Company, 1990.
- [24] F. Garzotto, P. Paolini, D. Schwabe, HDM-a model-based approach to hypermedia application design, *ACM Trans. Inform. Systems* 11 (1) (1993) 1–26.
- [25] F. Garzotto, L. Mainetti, P. Paolini, Navigation in hypermedia applications: modeling and semantics, *J. Organization. Comput. Electron. Commerce* 6 (3) (1996) 211–237.
- [26] D. Schwabe, G. Rossi, S.D.J. Barbosa, Systematic Hypermedia Application Design with OOHDM. *Hypertext '96 Proceedings*, ACM Press, 1996 116–128.
- [27] V. Balasubramanian, M. Turoff, 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 (1995) 241–250.
- [28] M. Bieber, S.O. Kimbrough, On generalizing the concept of hypertext, *Mgmt. Inform. Systems Quart.* 16 (1) (1992) 77–93.
- [29] M. Bieber, On integrating hypermedia into decision support and other information systems, *Decision Support Systems* 14 (1995) 251–267.
- [30] M. Thüning, J. Hannemann, J.M. Haake, Designing for comprehension: a cognitive approach to hypermedia development, *Commun. ACM* 38 (8) (1995) 57–66.
- [31] G. Salomon, A case study in interface design: CHI '89 Information Kiosk, in: R.M. Baecker, J. Grudin, W.A.S. Buxton, S. Greenberg (Eds.), *Readings in Human-Computer Interaction: Toward the Year 2000*, Morgan Kaufmann Publishers Inc., Los Altos CA, 1995, pp. 25–34.

- [32] WAIS, Inc., WAIS for UNIX, Chapter 1, Introduction, [http://www.wais.com/company/Tech\\_chap1.html](http://www.wais.com/company/Tech_chap1.html). 1995.
- [33] Information Dimensions, Inc. BASIS WEBserver™ Overview, <http://www.oclc.org/oclc/idi/basisweb/8012web.htm>. (1995).
- [34] Verity, Inc., Product Datasheet on Topic® WebSearcher, <http://www.verity.com/datasheet.html>. 1995.
- [35] H.V.D. Parunak, Toward industrial strength hypermedia, in: E. Berk, J. Devlin (Eds.), *Hypertext/Hypermedia Handbook*, : Intertext Publications/McGraw-Hill Publishing Co., New York, 1991, pp. 381–395.
- [36] R. J. Glushko, Transforming Text into Hypertext for a Compact Disc Encyclopedia, CHI '89 Proceedings, ACM Press (1989) 293–298.
- [37] B. Shneiderman, *Designing the User Interface*, Addison-Wesley Publishing Company Inc., Reading, MA, 1987.
- [38] P. Kahn, Visual cues for local and global coherence in the WWW, *Commun. ACM* 38 (8) (1995) 67–69.
- [39] R.M. Akscyn, D.L. McCracken, E.A. Yoder, KMS: a distributed hypermedia system for managing knowledge in organizations, *Commun. ACM* 31 (7) (1988) 820–835.
- [40] J.E. Conklin, Hypertext: a survey and introduction, *IEEE Comput.* 20 (9) (1987) 17–41.
- [41] J. Raskin, The Hype in Hypertext, *Hypertext'87 Proceedings*, ACM Press, (1987) 325–330.
- [42] New York University. New York University's Information Systems Department Handbook. <http://is-2.stern.nyu.edu/isweb>. (1996).
- [43] M. Rettig, Prototyping for tiny fingers, *Commun. ACM* 37 (4) (1994) 21–27.
- [44] T. Winograd, From programming environments to environments for designing, *Commun. ACM* 38 (6) (1995) 65–74.
- [45] V. Balasubramanian, Designing in the real world, *Interactions II*. 4 (1995) 15–17.
- [46] J. Nielsen, *Multimedia and hypertext: the internet and beyond*, AP Professional, 1995.
- [47] J. Nielsen, D. Sano, SunWeb: user interface design for sun microsystems' internal Web, *Proceedings of the Second World Wide Web Conference*, 1994.
- [48] T. Berners-Lee, R. Cailliau, A. Luotonen, H.F. Nielsen, A. Secret, The World Wide Web, *Commun. ACM* 37 (8) (1994) 76–82.
- [49] NCSA, The Common Gateway Interface, National Center for Supercomputing Applications, University of Illinois-Urbana Champaign. <http://hoohoo.ncsa.uiuc.edu/cgi/overview.html>. (1995).
- [50] Ohio State University. The HCI Bibliography Project by Gary Perlman. <http://www.cis.ohio-state.edu/~perlman/hcibib.html>. (1996).
- [51] J. Nielsen, *Hypertext and Hypermedia*, Academic Press Inc., New York, 1990.
- [52] N. Goodwin, Functionality and usability, *Commun. ACM* 30 (3) (1987) 229–233.
- [53] U. Rao, M. Turoff, Hypertext functionality: a theoretical framework, *Int. J. Human-Comput. Interaction* 2 (4) (1990) 333–358.
- [54] J.S. Brown, S.E. Newman, Issues in cognitive and social ergonomics: from our house to bauhaus, *Human-Comput. Interaction* 1 (1985) 359–391.
- [55] M. Bieber, C.J. Kacmar, Designing hypertext support for computational applications, *Commun. ACM* 38 (8) (1995) 99–107.
- [56] M. Bieber, *Hypertext and Web Engineering*. *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia*, ACM Press, 1998, 277–278.
- [57] K. Bennett, *Legacy Systems: Coping with Success*. IEEE Software, January, 1995, 19–23.