DESIGNING HYPERMEDIA APPLICATIONS

P. Balasubramanian

Tomas Isakowitz

Edward Stohr

Information Systems Department
Stern School of Business
New York University

<u>Working Paper Series</u>
STERN IS-93-40

Designing Hypermedia Applications

P. Balasubramanian [*] Tomás Isakowitz [†] Edward A. Stohr [‡]
Information Systems Department
Leonard N. Stern School of Business
New York, NY 10012

Abstract

We describe a step-by-step methodology for the design and construction of hypermedia applications and illustrate our approach using a small application. The Relationship Management Design (RMD) methodology begins with a data model of the application domain and proceeds through the design of the hypertext network, user interface and run-time dynamics finally concluding with the construction and testing of the target hypermedia system. Our ultimate objective is to use the RMD apporach as the basis for the construction of computerized tools to support the design and development of hypermedia applications.

1 INTRODUCTION

We propose a methodology for designing and constructing organizational hypermedia [Isa93] applications. Although many papers have been written about hypermedia [Con87, Bus45] and its applications [Min90], relatively few [Lan93, GPM93] address the design methodology issue. By design methodology we mean a systematic procedure to design and build hypermedia applications. A good design methodology should be efficient to use and should produce a hypermedia product that helps users obtain the information they need.

A good hypermedia system overcomes the inherent limitations of computer screens relative to paper (narrow window to the information and the need to manipulate mechanical paraphernalia). It also exploits the possibilities for rapid search and complex associations that can be provided in an electronic medium. The design of hypertext applications involves capturing and organizing the structure of a complex domain and making it clear and accessible to users [GPS93]. This is a time-consuming process often requiring the linking of many chunks of information residing in different media and connected in arbitrarily complex ways. A design methodology must first focus on how these connections are represented, manipulated and stored. Secondly, it must consider the dynamic interaction of the user with the hypertext (i.e. the "browsing semantics" [MS88, SF89]).

Once a satisfactory design has been determined, the hypertext still needs to be constructed. This is also a time-consuming and complex task even using a modern authoring system. It requires the manual development of the screen or window for each item of information followed by the construction of the links between items and the specification of any rules or procedures that define the dynamic aspects of the hypertext.

In this paper, we propose a methodology for both the design and construction of hypertext documents. In sections 1 through 4, we propose a seven-step methodology (see Figure 1) for designing hypermedia systems and illustrate it by means of an example that was developed first in the absence of a formal methodology and has since been redeveloped using the methodology. The methodology builds on previous research involving the use of data modeling in hypertext [FS90, Tom89, SLHS93, GPS93, AMY88]. In particular, we propose some extensions to the HDM2 data model developed in [GPM] to reflect the dynamic nature of links. We call the extended model the RMD (Relationship Management Design) model.

To construct the hypertext, we propose the use of a Relationship Management System (RMS) [IS92] that uses the output from the design stage to help generate and maintain the hypermedia system. As discussed in Section 6, the database capabilities of RMS can be used to input and maintain the information nodes and links.

[*] Doctoral Student in Information Systems
[†] Assistant Professor of Information Systems
[‡] Professor of Information Systems

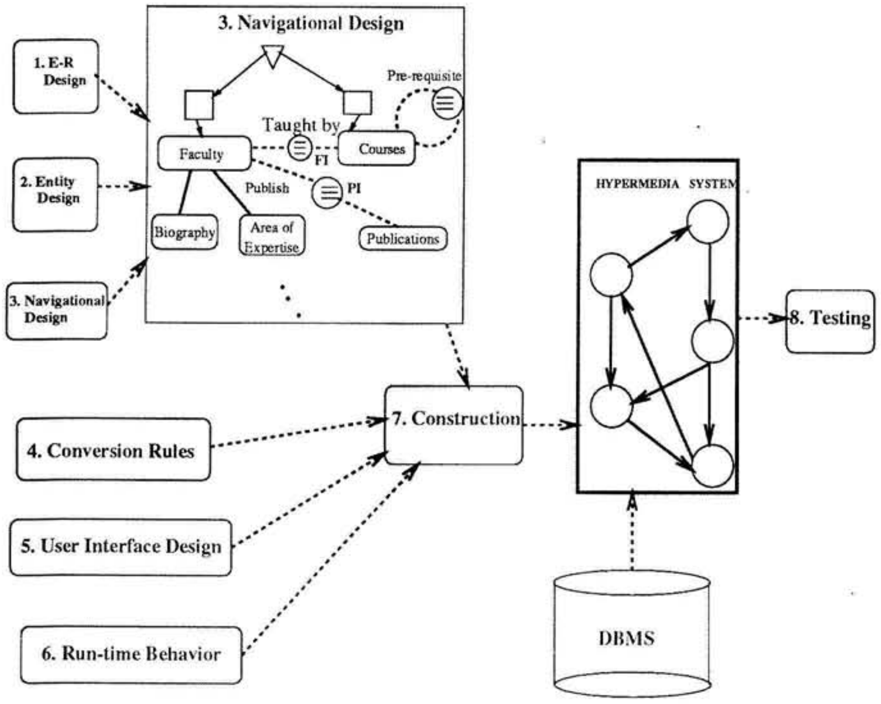


Figure 1: The RMD Design Methodology

2 THE HYPERMEDIA DESIGN PROBLEM

A hypertext or hypermedia system consists of chunks of information in the form of text, bitmapped images, sound, video and other programs. Each chunk of information is called a node. Nodes are connected to each other by links. Links are associated with a small part of the node called the anchor. When the user wants some information he/she activates the anchor and the system follows the associated link to access and display the target node. Since the text in a hypertext document can be browsed in more than one way, the access is non-linear and the potential exists for user confusion [Con87]. To design a hypermedia document, we need to decide how to divide the information domain into chunks, how the resulting chunks are to be linked, and how the user is to interact with the system.

In traditional hypertext design, application developers perform two tasks. The first task is to define the overall classes of information elements and navigational structure (authoring in the large) and the second task is to instantiate them (authoring in the small) [GPS93]. The small Information Systems department handbook application we use in this paper as a basis for discussion highlights the need for a formal methodology covering both of these steps. The handbook contains descriptions of programs and courses offered by the IS department and a list of the faculty members in the department.

Here's an excerpt from a report written by a member of the implementation team:

> *The first version of the system was developed along the same lines as the paper document. The handbook had a structure in the form of a Table of Contents, chapters, sections and sub-sections which was used as the basis for the design of the system. A segment of the book was converted into a hypertext system but even in the initial stages it became clear that the resulting system was not appealing... it had a strong hierarchical flavor to it... The system appeared to be a bunch of menus interspersed with pieces of text. The real problem came when we tried to go beyond the hierarchical book structure. To capture the hypertext structure, we had to create links between the hierarchical elements of the original document. As we had no specific rules on how these links were to be created, the design became unmanageable. It was tough to decide which elements were to be linked and which were to be left alone. This issue had*

Number of screens	96
Number of buttons/links	204
Elements of development time for one text screen	
Cut and paste text	12 - 15 min.
Add side panel	2 min.
Implement a button	1 min.
Implement a link	1 min.
Cosmetic changes (fonts, color, etc.)	4 min.
Number of buttons/links per screen	3 - 30
Total time to compose one screen	20 - 90 min

Table 1: Some Implementation Statistics

direct ramifications on the interface design and hence on the application.

To complete the design the team members developed a "design rationale" that included two different screen types and six different link types with different iconic representations for each. Even with a standardized approach that eliminated much decision-making, the implementation task was not trivial. The implementation took roughly two person months to complete over a four-month period. The system was developed using a popular PC-based hypermedia package. The two students who implemented the system were inexperienced in hypertext and had to learn the hypermedia package and its scripting language (which took an estimated 30 hours) as well as how to design a hypertext as noted above. The content of the hypertext document was provided in the form of an ASCII file and only simple graphics were used primarily in the initial screen which displayed the table of contents in an attractive graphic form.

Some (very approximate) statistics related to the implementation phase are shown in Table 1. In addition to the time spent on the above activities, an estimated 50% of the total development time was spent correcting mistakes and adding finishing touches.

From this exercise, it was apparent that we had much to learn about the art of hypermedia design. What visual elements are necessary? What should be linked to what? How does one prevent the user from being "lost in hyperspace"? etc. We also drew two conclusions from this exercise that guided the development of the methodology discussed in the remainder of this paper and helped shape the design and implementation of the second version of the handbook:

1. Capturing the underlying information structure of the application is important. Beyond that, one has to develop a secondary information structure, as an appendage to the application domain structure, that makes allowances for presentation requirements (multi-screen nodes, etc.) and for the dynamics of the user interaction.

2. It was evident from the development experience that it was hard to look after many application details in a consistent and efficient manner. In particular, it was difficult to maintain consistency between screens and links of the same type. A semi-automated system that would lead the developer through the design, screen-by-screen and link-by-link would be helpful.

A number of existing approaches to hypertext design have been influenced by the database design field. Database models are useful abstractions in database applications but the peculiarities of hypermedia demand the development of new design models appropriate for hypermedia applications.

Several hypertext-oriented data models have been proposed in the literature that provide guidelines for the design of hypermedia applications [GPS93]. A specific database model is implicit in gIBIS [CB89], a tool for exploratory policy discussions. Here, the nodes and links are determined by the application domain (policy and design discussions). This has a well-defined theory: issues, arguments and positions are represented by node types that are connected by nine link types (responds-to, supports, objects-to etc.).

In a more general approach, [SLHS93] a semantic object-oriented database paradigm is used to describe the datamodel and a hyperbase management system, HB1, employing a semantic network database management system is used to manage the physical data storage and link information.

The dynamics of the user interaction with the hypertext also needs to be designed. The Trellis model [FS90] suggests that hypertext applications be modeled using Petri nets. The Petri-net formalism provides a convenient mechanism for specifying the browsing semantics for the application desired. Other papers treating hypertext representation schemes and the dynamic aspects of hypertext include Tompa [Tom89], [CG88], HAM [CG88] KMS [AMY88] and HDM [GPM93].

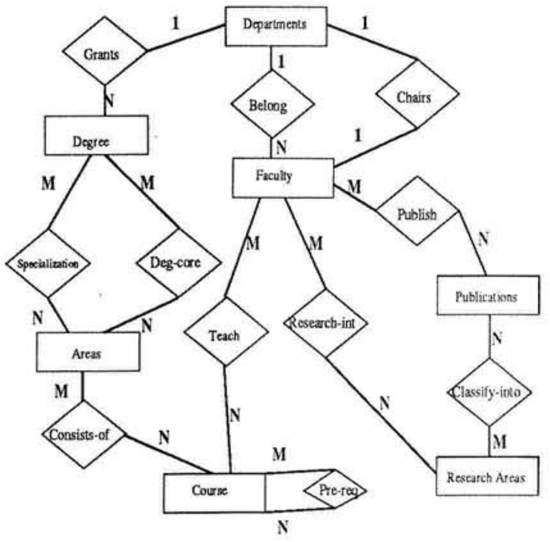


Figure 2: E-R diagram for the MBA Handbook

3 RELATIONSHIP MANAGEMENT DESIGN MODEL (RMD)

The basic structure behind the RMD model is the entity-relationship (E-R) model [Che76]. The E-R model is used because it is quite intuitive and familiar to many designers. Figure 2 shows the E-R diagram for part of the IS handbook.

The E-R diagram conveys information about the structure of the domain. However, it does not specify all the navigational and access mechanisms. The RMD methodology extends the initial E-R diagram into a full-fledged hypermedia design diagram.

RMD borrows heavily from the constructs defined in HDM [GPS93] and HDM2 [GPM]. In HDM2 entities are hierarchical structures composed of components. The root node represents the highest level of the entity and each component has a part_of relationship with its parent. The root node and components have associated *perspectives*. For example, a component may have different representations in different media - video, sound etc. A unit corresponds to a node in the node-link data model. It is the smallest amount of information that is accessible by the user and can be the root node of the entity, a component or a perspective. Each unit has its own display window. HDM2 provides two link types called connections. *Structural* connections define the structure of an entity type by binding together its components and their *perspectives*. *Applicative* connections describe relationships among entity types by connecting compo-

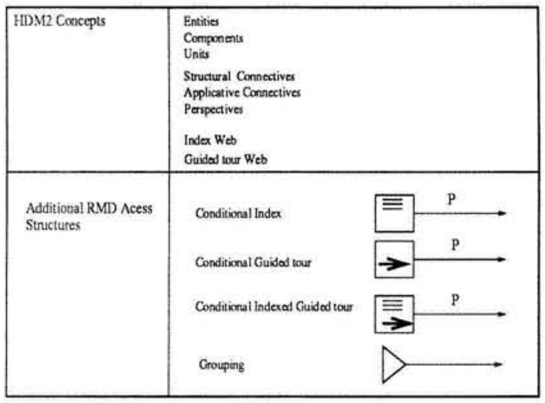


Figure 3: Design Concepts and Additional Notations

nents of entities belonging to different entity types. In addition, HDM2 provides two kinds of access structures: index webs and guided tour webs. An index is an access mechanism which allows the user to choose a destination node; physically, an index may be a set of buttons or a pop-up menu. A guided tour is an ordered list of information units created by the author of an application and used by the reader. Guided tours provide rigid paths to visit nodes that the author considers important [GPM93, TSH86]. Some of the actions that can be taken from a node in a guided tour are activate, next, previous, first, last, suspend, and resume. [GPM93].

We now describe the additional elements of the RMD model and explain the concepts using an example. Figure 3 shows the various elements we have added to HDM2. The first element, denoted by a rectangle with lines enclosed inside and a predicate(P), is a *conditional index*. A conditional index with a predicate P is an index to the subset of the instances satisfying the predicate P. A conditional index without a condition is considered to have the *true* predicate, and corresponds to the notion of an index in HDM2. For example if we want an index into all the programs that are offered in the department we connect a group index to the entity program without any conditions (see Figure 5).

Every datamodel has to be translated eventually into a node-link data model and then implemented. In our case each entity is translated to a collection of nodes and each relationship is translated to one or more links. When the user attempts to navigate from one entity instance to the other via a *one-to-*

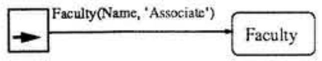


Figure 4: A guided tour example

many applicative relationship **R**, a conditional index is activated. The index contains a list of destination node id's whose attribute values satisfy the predicate representing the relationship **R**. Since the node id's in the index are conditional upon the nodes that satisfy the predicate we call these indices *conditional indices*.

The next two constructs shown in Figure 3 are for guided tours. Our representation deals with a restricted version of guided tours that includes only instances of the same entity type. The restriction is needed to enable automatic construction of these guided tours (see section 5). More general guided tours with nodes of different entity types cannot be specified using our graphic conventions.[1] Conditional guided tours are denoted by a crossed circle and a predicate. A Conditional guided tour specifies a sequence of nodes that satisfy a predicate.

The user can browse the nodes of a guided tour using the next and previous arrow key. For example, Figure 4 denotes a guided tour of all associate professors. A guided tour with no predicate, stands for a conditional guided tour with predicate *true*.

The next construct specifies a conditional indexed guided tour; this is similar to a conditional guided tour with the addition of an index into the nodes in the tour. An indexed guided tour with no predicate is a conditional indexed guided tour with predicate *true*.

The last construct, denoted by a triangle (see Figure 5), is used to denote groupings to provide users with a means to access other access structures such as indices or guided tours.

Before we describe the design methodology itself, we will discuss additional RMD examples. Figure 5 shows a portion of the RMD diagram of the handbook application. In this diagram entities are shown as rectangles and relationships are shown as lines. Structural relationships are drawn with solid lines and applicative relationships with dashed lines. Attributes of entities are shown in italics inside ellipses. Only the key attribute is shown in the RMD diagram, the rest of the attributes are described in the entity diagram defined in step 1 of our methodology. Figure 5 shows a conditional index between faculty and courses with predicate *Teaches(name, course)*. The word **CI** just outside the circle refers to the course index that is created/activated when we move from the faculty node to a course node and **FI** refers to the faculty index that is created/activated during the traversal from course to faculty node. In essence, these two indices represent the *many-to-many* relationship between *faculty* and *course* shown in the E-R diagram. The notation used in the figure abbreviates these two separate indices, one into faculty with predicate taught-by and another into courses with predicate teaches. An implementation of the RMD diagram in Figure 5 is shown in Figure 6. The destination nodes of the link *teaches* are just those courses taught by *Shocken*; not all courses taught by faculty in the IS department. Moreover, the user can choose the destination node from an index.

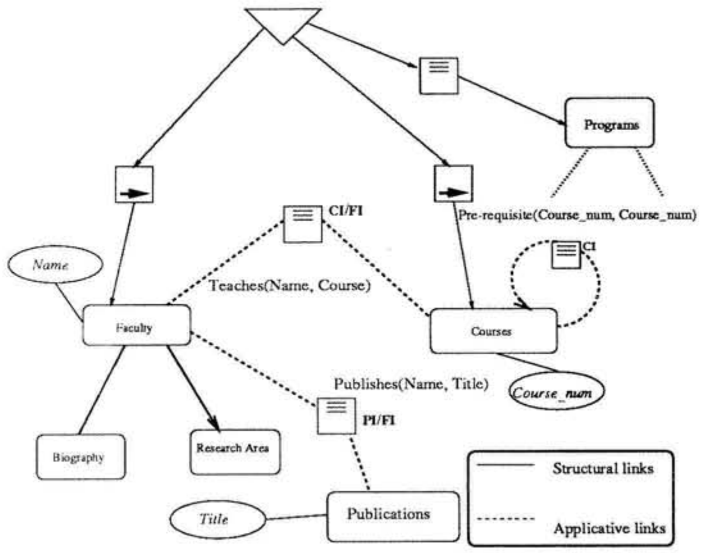


Figure 5: Part of an RMD Diagram

In Figure 5 we also show the grouping mechanism that we have used to implement the handbook application. Access into the *faculty* and *course* information is provided via guided tours; access into *programs* by means of an index. On choosing the guided tour to the *faculty* entity, the user can move from one faculty member to the next. From the *faculty* node there is a conditional index to *courses*. This allows the user to move from the *faculty* entity to the courses taught by that faculty member, as indicated by the predicate *Teaches*. Had we provided an *indexed guided tour* to the courses taught by a faculty the user would have been able to choose which *course* node to visit and from there he/she could use the next (previous) arrows to see the next (previous) courses taught by the same faculty member.

[1]Rather, they are specified by stepping through the nodes manually and recording the path taken.

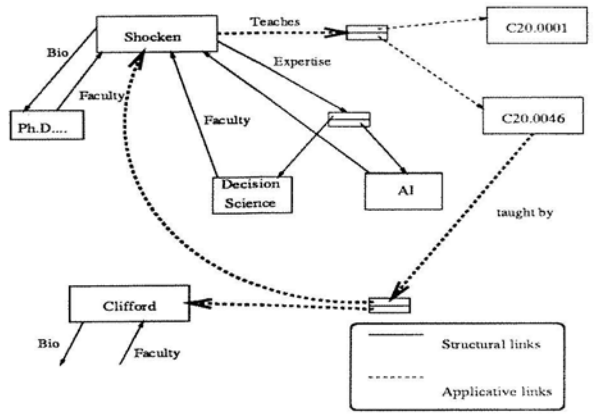


Figure 6: Implementation of the Handbook

Step	Description
S1	E-R Design
S2	Entity Design
S3	Navigational Design
S4	Conversion Protocol Design
S5	User-Interface Design
S6	Run-Time Behavior Design
S7	Construction

Figure 7: The steps of the RMD Methodology

4 THE RELATIONSHIP MANAGEMENT DESIGN METHODOLOGY

In this section, we describe the RMD methodology for designing and constructing hypermedia applications. This methodology has been developed to support computer-aided hypermedia development (CAHD).

In order to illustrate our concepts we discuss the design of the revised version of the Information System Department's MBA Handbook. In a future study we plan to evaluate this design using principles reported in a Bellcore study [ERG+89].

The design methodology we propose consists of seven steps listed in Figure 7 which are to be iterated as needed.

S1: E-R Design.

We first determine the aspects of the domain that are to be supported by the application and represent these with an E-R diagram [EN90]. Since many designers have experience with E-R diagrams we can build on such experience to provide a familiar environment for the design of hypermedia documents. Following the HDM nomenclature [GPM93, GPM, GPS93], we call these relationships *applicative*. The E-R diagram for the IS handbook application was shown in Figure 2. The diagram does not indicate the ways in which users may navigate through the information space; rather it is a design of the entities and relationships among them. This helps determine the entities and the applicative links to be present in the application.

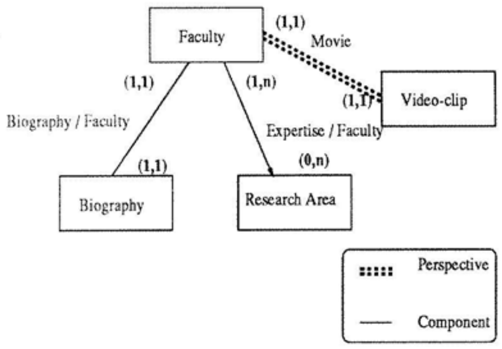


Figure 8: Entity diagram for faculty

S2: Entity Design.

We now proceed to design how the information content of domain entities is to be divided into information units following the HDM2 approach. This results in *Entity Diagrams* that depict the structural and perspective relationships of the entities in the application. Figures 8 and 9 illustrate the hypermedia representation for the *Faculty* and *Programs* entities. The *Faculty* entity has one perspective link to a video clip and two structural links, *biography*, which is one-to-one, and *expertise* which is *one-to-many*. Recall that structural links relate the values of attributes of an instance of an entity. A one-to-one structural link originates from a single-value attribute; a one-to-many link from a multivalued attribute.

S3: Navigational Design.

The next step is navigational design during which stage we combine the E-R diagram and the entity diagram to produce an RMD diagram (Figure 5). Navigational design is important because it will help reduce disorientation [Nie90]. RMD differs from HDM2 in that applicative relationships are replaced by *conditional indices* as are one-to-many structural relationships. In HDM2, plain indices are used to show such relationships. Our notation helps automate the process of building indices.

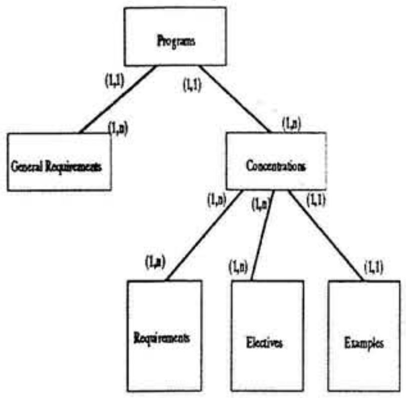


Figure 9: Entity diagram for program

S4: Conversion Protocol Design.

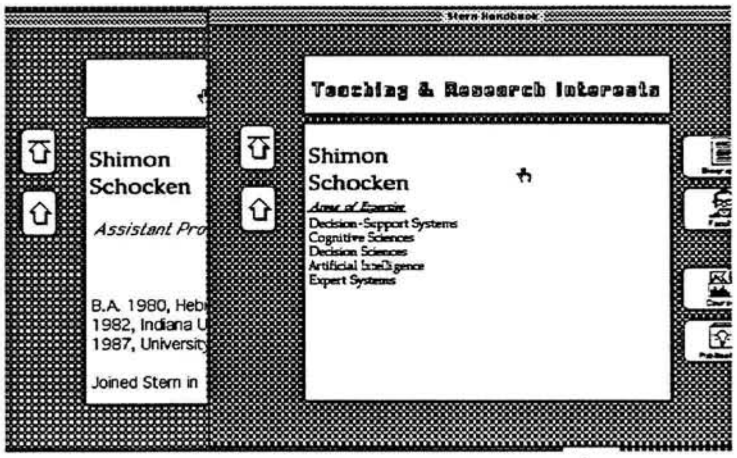
The next step is to specify how the RMD diagram is to be realized in the target hypermedia application platform. This is done via a set of *conversion rules* – explained in section 5 – that determine the implementation of relationships and access structures. The actual implementation is platform specific. Figure 10 shows how the *expertise one-to-many* structural link is implemented in the IS handbook. Alternatively, one might opt to use a pop-up box to implement a *one-to-many* relationship as shown in Figure 11.

The conversion rules are dependent on the target hypermedia application. However, all hypermedia platforms have the ability to process the plain-vanilla *nodes-&-links* data model. Hence, rules that specify a conversion of RMD entities and relationships into nodes and links are general enough to be processed by most hypermedia platforms. We present such rules in section 5. They transform the combined diagram of Figure 5 into a web of nodes and links as indicated in Figure 6.

S5: User-Interface Design

This stage involves the design of screen layouts for every object in the RMD model. This includes button layouts, appearance of nodes and indices and location of navigational aids. User-Interface guidelines [Shn87, SK92] and user-involvement are crucial in this step.

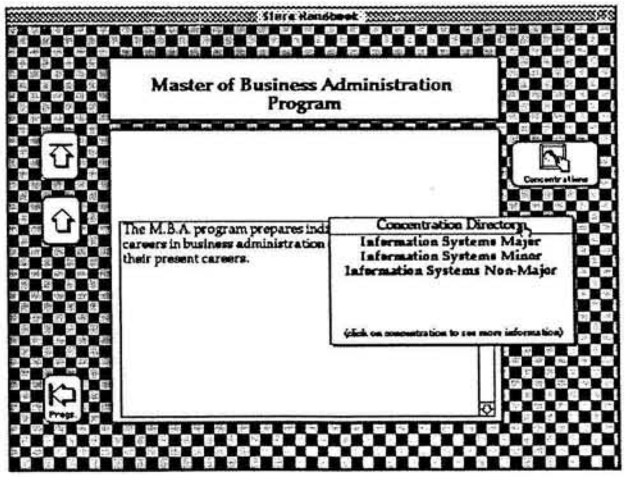
We used a set of simple guidelines in our implementation. All text is shown in a central portion of the screen, with no embedded buttons. Navigation buttons (go to the main menu, go back etc.) are shown on the left hand side of the screen. Buttons corresponding to structural links are placed on the top right hand corner of the screen. Buttons corresponding to applicative links are shown in the bottom right corner. Bi-



Shimon
Schocken
Assistant Pro
B.A. 1980, Heb
1982, Indiana U
1987, University
Joined Stern in

Teaching & Research Interests
Shimon
Schocken
Areas of Expertise
Decision-Support Systems
Cognitive Sciences
Decision Sciences
Artificial Intelligence
Expert Systems

Figure 10: One possible implementation of a one-to-many structural link.



Master of Business Administration Program

The M.B.A. program prepares indi
careers in business administration
their present careers.

Concentration Directory
Information Systems Major
Information Systems Minor
Information Systems Non-Major
(click on concentration to see more information)

Figure 11: An alternative implementation of a one-to-many structural link.

directional arrows placed in the lower middle portion of the screen enable navigation among siblings, i.e. next/previous faculty member, next/previous course.

In the IS handbook application, screen designs were done manually as HyperCard backgrounds that apply to all nodes of a given class or entity type. It is important to realize that the buttons to be placed within each screen are derived from the RMD diagram. For instance, each index access into an entity could conceivably result in a *return-to-index* button. The problem of properly designing such screens in the context of multiple access structures is a challenging one, which we are only starting to address here.

S6: Run-Time Behavior Design

This step involves the design of the application's run-time behavior by specifying the semantics of link traversal and how to obtain the information to be presented. It is at this stage that designers determine the dynamic or static nature of links and nodes. Traversal of a static link involves accessing the endpoint of an existing link, whereas traversal of a dynamic link involves extracting information not present in the link from other sources (perhaps from an outside application, e.g. an on-line database) to compute a link's destination node. (The node in turn can also exhibit dynamic behavior) Dynamic links support domain inter-operability because they enable access to systems outside the hypermedia application.

In the IS handbook application, HyperTalk scripts were manually coded for the buttons in all background cards. Some applicative links are semi-dynamic. For example, the conditional applicative index *teaches* that produces an index of all courses taught by a faculty member is resolved on the fly by querying a database. (We call this semi-dynamic because at this stage of the prototype implementation, the database is internally coded in HyperCard.)

S7: Construction and Testing

During this stage, which is not part of design, but of implementation, the hypermedia application is constructed by populating the derived node-&-link datamodel with domain data, as dictated by the RMD conversion protocol and the run-time behavior designs. Figure 1 illustrates this last step, which involves an *RMD construction* process.

In the IS handbook application, this step was performed manually by pasting the text from an available ASCII version of the IS handbook into instances of the HyperCard cards designed in stages **S5** and **S6**.

Clearly, there is enough information in the RMD diagram, screen design guidelines, conversion rules and run-time behavior specifications to automate this compilation process. We hope to report on such a tool in the near future.

After construction, the application is tested and the process (from stage **S1** on) is iterated as many times as needed.

5 CONVERSION RULES

In this section we informally describe the conversion rules we utilized in translating the RMD diagram into a a web of nodes and links. There are rules for converting every kind of relationship and access structure present in the RMD diagram into a web of nodes and links. We illustrate some of these rules in this section.

As shown in Figure 12-1, a *one-to-one* relationship (applicative or structural) between unit types U1 and U2 in the RMD diagram results in the creation of two links. One link enables the user to navigate from N1 (the node representing a unit of type U1) to N2 representing a unit of type U1). The other link takes the user from N2 to N1. The label L/L^{-1} indicates that the relationship is bi-directional, with each direction appropriately labeled. The next example, portrayed in Figure 12-2, shows the implementation of a *one-to-many* structural relationship. A pop-up intermediary node, as opposed to a floating window, is used to stress that the information is strongly related to the current entity.

Figure 13 illustrates the conversion rule for conditional indices, which are implemented using a special node (the index node) with links to the root node Nj of the entity E satisfying the predicate P.

A guided tour is implemented with an introductory node that explains what the guided tour is about. From this node there is a link to the first node N_1 satisfying P_1, from N_1 to the second node N_2 and so on.

Group access structures provide entry points to the information in the hyperbase. The group access structure in Figure 14 is implemented as a separate node with links to introductory nodes for the index and the guided tour.

The links we described so far were explicit in the RMD diagram itself, we call these direct links. The RMD framework also handles automatically *derived links*. The general rule for derived links is shown in Figure 15. This derivation rule propagates relationships along structurally related components. Its application is illustrated in Figure 16. The first example

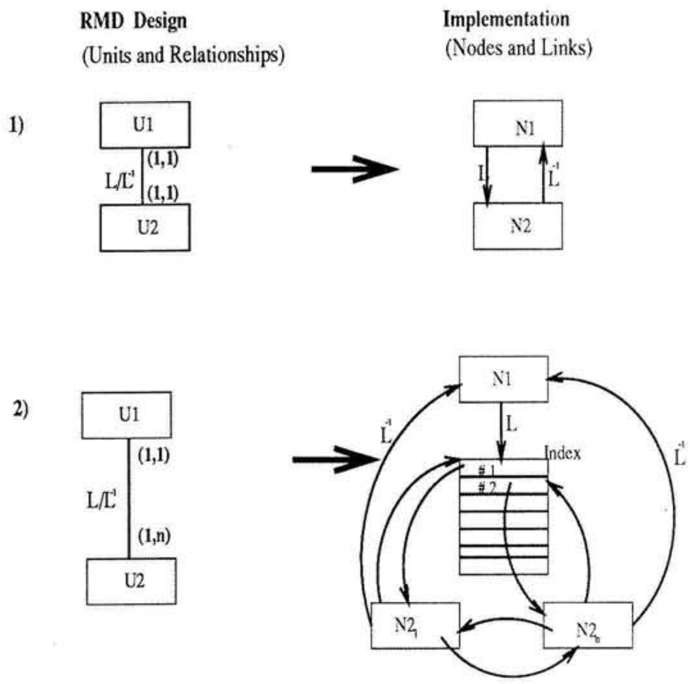


Figure 12: RMD diagram conversion principles for direct links

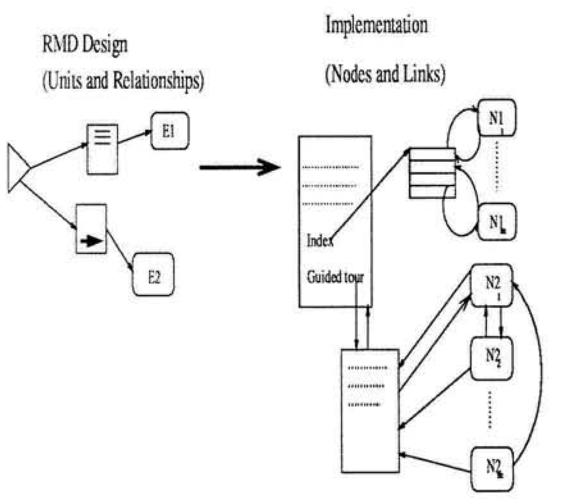


Figure 14: Conversion Rules for Access Structures

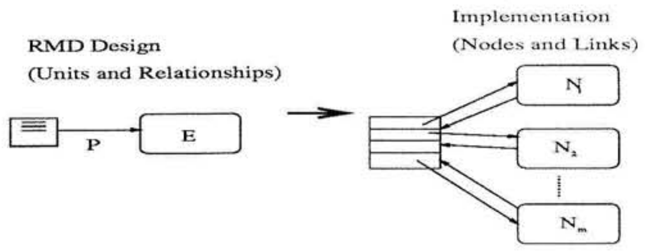


Figure 13: Conversion Rules for Indices

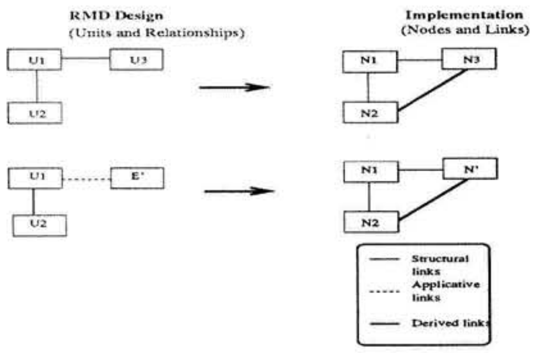


Figure 15: Entity diagram conversion rules for deriving links

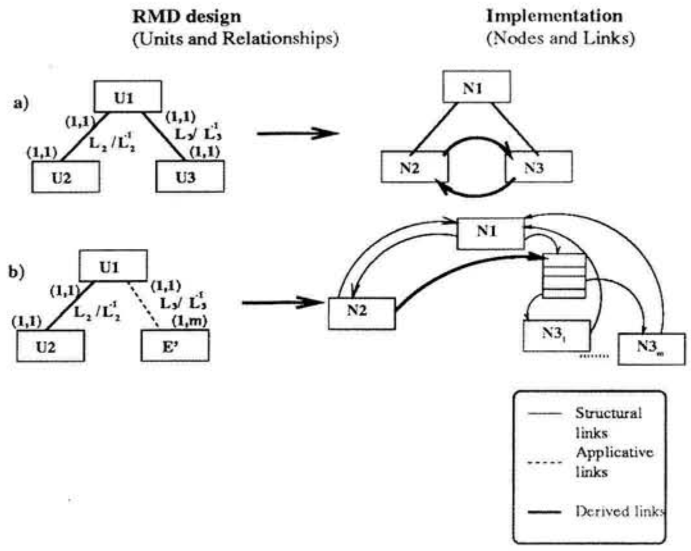


Figure 16: Examples of derived links

illustrates a *one-to-one* relationship between U1 and U2 and between U1 and U3. The derivation rule is applied twice, once to U2 and once to U3, resulting in the creation of two links: one from N2 to N3 and another from N3 to N2.

In the second example we have a *one-to-one* structural relationship between U1 and U2 and a *one-to-many* applicative relationship between U1 and U' (actually the relationship is between the entity E – of which U1 is a part – and E' – of which U' is a part). In this case the derived links are between N2 and $N3_1$ … $N3_m$, where $N3_1$ … $N3_m$ are the roots of the appropriate entity instances of the type E'. This time the index is implemented as a floating window node since it denotes an applicative link.

Applying these link conversion rules will result in several links being created. However, the number of links will not expand exponentially because each node is linked only to its parent (and index in the case of *many-to-many* relationships) and to the next sibling.

The conversion rules that we have outlined here were used in the implementation of the IS handbook. The designer can substitute these rules with different ones to reflect his or her own judgment. For example, in Figure 16 the designer may insist that every node that is a destination of a conditional index have a link back to its index.

6 RELATIONSHIP MANAGEMENT SYSTEM

In the previous section we described a methodology for designing hypermedia applications. This methodology involves dealing with several kinds of relationships in the application domain. In this section we give a very brief outline of the role of a relationship management system (RMS) that supports the establishment, maintenance and use of such relationships especially among independent applications. Implementation of *one-to-many* and *many-to-many* relationships can be handled in three ways:

Static : Indices are built into the application by the application developer and do not change. In our example the areas of expertise of a faculty member are stored as static indices (**EI**) (see Figure 6).

Dynamic (semi-interoperable) : Indices are created on the fly but the databases used are internal to the development environment. By internal we mean that the application itself stores the information as opposed to storing it in a general purpose database management system. This is the way we have implemented all our dynamic indices. For example the faculty index (**FI**) in Figure 6 is created using a database implemented within Hypercard.

Dynamic (fully-interoperable) : In this case, indices are created at run time by accessing information from an external database. For example to obtain all publications by *Schocken*, the system would execute the following query to an external database:

$$\pi_{Title}(\sigma_{Faculty\_id = \text{"Shocken"}}$$

$$(Faculty\_Publications \bowtie Publications))$$

and build an index of publication titles. The user can choose to select any publication from this list.

The above is a very brief outline a RMS system. In a future paper, we will report our progress in developing a more general RMS.

7 CONCLUSION AND FUTURE WORK

We have proposed a methodology for the design and implementation of hypermedia applications and illustrated its use through an actual example. The existence of a well-defined methodology based on a data model of the application domain facilitated the development process and helped produce a more complete and consistent design in the sample application. However, the development of an effective hypermedia design approach will require much further research. In this regard we are currently investigating a RMD language definition and an RM schema conversion language. We wish to develop an RMD editor to assist users in drawing the RMD diagrams, an RMD compiler to apply the various rules and build the application and a screen design tool to customize the screens for each entity. We also plan to build other applications and to conduct experiments to evaluate the design methodology.

Acknowledgments

This research was supported in part by a grant from Apple Computer Inc. We also wish to thank the following people for their help in implementing the department of information systems handbook in hypercard: Terrance Chen, Hon-Fun Kum; Angeline Wong for loading the information into the hypercard implementation; Alex Bronstein and Rama Krishnan for their implementation of the first version of the handbook.

References

[AMY88] Robert M. Akscyn, Donald L. McCracken, and Elise A. Yoder. KMS: A Distributed Hypermedia System For Managing Knowledge In Organizations. *Communications of the ACM*, 31(7):820–835, July 1988.

[Bus45] Vannevar Bush. As we may think. *Atlantic Monthly*, pages 101–108, July 1945.

[CB89] Jeff Conklin and Michael L. Begeman. gIBIS: A Tool for All Reasons. *Journal of the American Society for Information Science*, 20(3):200–213, 1989.

[CG88] B. Campbell and J. M. Goodman. HAM: A general Purpose Hypertext Abstract Machine. *Communications of the ACM*, July 1988.

[Che76] P. P. Chen. The Entity Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[Con87] Jeff Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, 20(9):17–41, September 1987.

[EN90] Ramez Elmasri and Shamkant Navate. *Fundamental of Database Systems*. The Benjamin/Cummings Publishing Company, second edition, 1990.

[ERG+89] Dennis E. Egan, Joel R. Remde, Louis M. Gomez, Thomas K. Landauer, Jennifer Eberhardt, and Carol C. Lochbaum. Formative Design-Evaluation of Superbook. *ACM Transactions on Office Information Systems*, 7(1):30–57, January 1989.

[FS90] Richard Furuta and P. David Stotts. The Trellis Hypertext Reference Model. In Judi Moline, Dan Beningni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 83–93, Gaithersburg, MD 20899, March 1990. National Institute of Standards and Technology. NIST special publication 500-178.

[GPM] Franca Garzotto, Paolo Paolini, and Luca Mainetti. Navigation in Hypermedia Applications: Modelling and Semantics. *Journal of Organizational Computing*. (forthcoming).

[GPM93] Franca Garzotto, Paolo Paolini, and Luca Mainetti. Navigation Patterns in Hypermedia Data Bases. In *Proceedings of the 26th Hawaii International Conference on System Sciences, Vol. III*, pages 370–379. IEEE Computer Society Press, January 1993.

[GPS93] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. HDM - A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Office Information Systems*, (1):1–26, January 1993.

[IS92] Tomás Isakowitz and Edward A. Stohr. Hypertext-based Relationship Management for DSS. Working paper IS-92-22, Center for Research in Information Systems, New York University, Information systems Department, New York, NY 10003, 1992.

[Isa93] Tomás Isakowitz. Hypermedia, Information Systems and Organizations: A Research Agenda. In *Proceedings of the 26^{th} Hawaii International Conference on System Sciences, Vol. III*, pages 361–369. IEEE Computer Society Press, January 1993.

[Lan93] Danny B Lange. Object-Oriented Hypermodeling of Hypermedia Supported Information Systems. *Proceedings of the 26th Hawaii International Conference on System Sciences*, III:380–389, 1993.

[Min90] R.P. Minch. Applications and Research Areas for Hypertext in Decision Support Systems. *Journal of Management Information Systems*, 6(3):119–138, Winter 1989-90.

[MS88] G. Marchionini and Ben Shneiderman. Finding Facts vs. Browsing Knowledge in Hypertext Systems. *IEEE Computer*, pages 70–80, January 1988.

[Nie90] Jakob Nielsen. The Art of Navigating Through Hypertext. *Communications of the ACM*, 33(3):297–310, March 1990.

[SF89] P. David Stotts and Richard Furuta. Petri net based Hypertext: Document Structure with Browsing Semantics. *ACM Transactions on Information Systems*, 7(1), January 1989.

[Shn87] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading, Massachusetts, 1987.

[SK92] Ben Shneiderman and Greg Kearsley. *Hypertext Hands-On! An Introduction to a New Way of Organizing and Accessing Information*. Addison-Wesley, Reading, Massachusetts, 1992.

[SLHS93] John L. Schnase, John J. Leggett, David L. Hicks, and Ron L. Szabo. Semantic Data Modeling of Hypermedia Associations. *ACM Transactions on Information Systems*, 11(1):27–50, January 1993.

[Tom89] Frank Wm. Tompa. A Data Model for Flexible Hypertext Database Systems. *ACM Transactions on Information Systems*, 7(1), January 1989.

[TSH86] Randall H. Trigg, Lucy A. Suchman, and Frank G. Halasz. Supporting Collaboration in Notecards. In *Proceedings of the Conference on Computer Supported Cooperative Work, Austin, TX*, pages 1–10, Dec. 3-5, 1986.