

# RMM: A Methodology for Structured Hypermedia Design

Tomás Isakowitz<sup>1</sup> Edward A. Stohr<sup>1</sup> P. Balasubramanian<sup>2</sup>

May 3, 1995

## Abstract

Hypermedia application design differs from other software design in that it involves navigation as well as user-interface and information processing issues. We present the Relationship Management Data model (RMDM) and the Relationship Management (RMM) methodology for the design and development of hypermedia applications. The seven steps of the methodology lend themselves to computer support, paving the way for a computerized environment to support the design and development of hypermedia applications. This article focuses on design activities, which are addressed within the first three steps of the methodology.

## Introduction

Hypermedia development, especially on a commercial scale, often involves teams of developers who need to be managed and coordinated over an extended period of time. Formal systems development and project management techniques are needed to ensure that the hypermedia product meets its objectives and is completed on time and within budget. However, traditional software industry techniques must be modified to accommodate new requirements. Hypermedia projects differ from traditional software development projects in several critical dimensions. First, as Streitz notes in his sidebar (in this issue), hypermedia projects may involve people with very different skill sets: authors, librarians, content designers, artists, and musicians, as well as programmers. Second, the design of hypermedia applications involves capturing and organizing the structure of a complex domain and making it clear and accessible to users [9]. Third, multimedia aspects of hypermedia applications raise numerous difficulties [13]. Hypermedia design is therefore a challenging process that is currently more of an art than a science. Finally, the need for prototyping (see the paper by Nanard and Nanard in this issue) and intensive testing with users is even more pronounced in

---

<sup>1</sup>Information Systems Department, Leonard N. Stern School of Business, New York University, New York, NY 10012

<sup>2</sup>Management Information Systems Department, School of Management, Boston University, Boston, MA 02215

hypermedia development than it is with traditional software because user tolerance to errors in hypermedia applications is very low.

In this article we propose the relationship management methodology (RMM) for the design and construction of hypermedia applications. The name “*relationship management*” stems from our view of hypermedia as a vehicle for managing relationships among information objects.

The class of applications for which RMM is most suited exhibits a regular structure for the domain of interest, i.e., there are classes of objects, definable relationships between these classes and multiple instances of objects within each class. Many hypermedia applications satisfy this requirement. Examples include product catalogs and hypermedia front-ends to traditional database or “legacy” applications. Since many hypermedia applications in this class have volatile data that requires frequent updating, some means to routinize and automate both the initial development and subsequent update process is needed.

Table 1 illustrates the usefulness of the RMM approach to design and development of hypermedia applications. The two axes representing the structure and volatility of the information are really continuums rather than discrete dichotomies. Applications in the two domains we mentioned above – product catalogs and hypermedia front-ends to databases or legacy applications – have high structure and high information volatility and the RMM methodology is particularly appropriate. At the opposite end of the spectrum, an artistic work may not have a readily discernible structure and usually remains unchanged over time. In this case, RMM is not applicable. Applications that have high structure and remain unchanged over a long period of time can benefit from the RMM methodology during the design and construction phases but do not require much maintenance so that the updating problem is relatively unimportant and the advantage to be gained by the RMM approach is not as pronounced. Finally, applications that have irregular (or dynamic) structures and high volatility may gain little from the use of the RMM approach. In this case, however, we reserve judgment as it is possible that some parts of the domain may be structured and that the problem of high information volatility can at least be partially addressed.

A number of commercial products, e.g., *Documentum* by Documentum Inc., *PDM* by Xyvision Inc., *RDM* by Interleaf, and *SGML Server* by Information Dimensions, use an approach similar to RMM, in that they provide flexible access mechanisms to documents that are stored in a database. The approaches used in these products are proprietary and have not been reported in the research literature. Moreover, these systems do not provide support for the design process, which is the

| <b>Volatility of Information</b> |             |  |  |
|----------------------------------|-------------|--|--|
|                                  | <b>Low</b>  | <b>High</b>                                    |  |
| <b>Structure</b>                 | <b>High</b> | Medium usefulness<br>[e.g., Kiosk Application] | High usefulness<br>[e.g., Product catalog, DBMS interface] |
|                                  | <b>Low</b>  | Not useful<br>[e.g., Literary work]            | Low usefulness<br>[e.g., Multimedia news service]          |

Table 1: Usefulness of RMM Approach

focus of RMM.

The paper is organized as follows. In the next section we present the relationship management data model which provides the foundation of the methodology. Thereafter, we introduce a seven-step methodology for the design and construction of hypermedia projects. The final section of the paper provides conclusions and a brief overview of future work.

## Relationship Management Data Model (RMDM)

A data model is a set of logical objects used to provide an abstraction of a portion of the “real world.” Data models are necessary to express an application’s design. A number of researchers have developed data models for hypertext systems, e.g., [18], [24], Dexter [12], [11], HB1 and HB3 [16, 20] and Trellis [23]. However, it is important to differentiate a hypertext system from a hypertext application. The former is an environment that facilitates the creation of the latter. A data model for a hypertext system ([1], [7], [24], [20]) details its internal architecture but is of little value in modeling hypermedia applications. This is because describing the layout of a general purpose engine is quite different from modeling an application domain; a different kind of data model is needed for this purpose. In our case, RMDM provides a language for describing the information objects and the navigation mechanisms in hypermedia applications.

Database models are useful abstractions in database applications, but the peculiarities of hypermedia, in particular the navigation aspect, require new models. Garzotto, Paolini and Schwabe’s HDM data model [9] (see also the article by Garzotto, Paolini and Mainetti in this issue) is appropriate for describing the structure of the application domain, and we base our data model on HDM and its successor HDM2 [8]. HDM and HDM2 describe representation schemes but provide little information on the procedures for using those representations in the design process, i.e., they do not describe a hypermedia design and development methodology. Lange [15] and Schwabe and

Rossi (see their sidebar in this issue and [22]) study object-oriented approaches to hypermedia design. Recent works by us [2] and by Garzotto, Mainetti and Paolini [10] focus on this problem. The present paper represents an update and expansion of [2]. Our methodology differs from that presented in [10] in several dimensions including the recommended sequence of steps, additional access structure formalisms, increased emphasis on graphic representations and a more detailed, step-by-step, procedure for hypermedia design and development. The proceedings of the two recent hypermedia design workshops [21, 14] discuss various other issues arising in hypermedia design.

INSERT FIGURE 1 ABOUT HERE

We now describe the Relationship Management Data Model (RMDM) which is the cornerstone of the RMM methodology. Figure 1 shows RMDM's modeling primitives. In the upper part of the Figure are the *domain primitives*, which model information about the application domain. *Entity* types and their attributes represent abstract or physical objects, such as *person* or *bank account*. *Associative relationships*, which can be *one-one* or *one-many*, describe associations among different entity types. As in database modeling, a many-many relationship is factored into two one-many relationships.

Because entities may consist of a large number of attributes of a different nature (e.g., salary information, biographical data, photographs), it may be impractical or undesirable to present all of the attributes of an entity instance at once. Thus, attributes are grouped into *slices*. For example, a *person* entity with attributes *name*, *age*, *picture* and *biography*, may have a *General slice*, containing *name*, *age* and *photograph* and a *Biography slice*, with *name* and *biography*. Hence, each instance of the entity *person* will be presented by two slices, and, if the application supports it, a user may choose which one to view. The graphical notation for slices – meant to resemble a *pizza slice* – appears in the middle of Figure 1.

INSERT FIGURE 2 ABOUT HERE

Navigation is supported in RMDM by the six access primitives shown at the bottom of Figure 1. The uni-directional and bi-directional links are used to specify access between slices of an entity. It is important to stress that these links can only be used to navigate within the boundaries of an entity. RMDM supports navigation *across* different entities via *indices*, *guided tours* and *groupings*. An index acts as a table of contents to a list of entity instances, providing direct access to each listed item. A guided tour implements a linear path through a collection of items allowing the user to move either forward or backward on the path. There are a number of useful variations on guided tours. For example, a “circular guided tour” links the last element back to the first; a “guided tour with return to main” has a distinguished node that contains information about the guided tour itself (e.g., “this is a guided tour of faculty homepages”), and is both the starting and ending point of the tour; and a “guided tour with entrance and exit” has different entrance and exit nodes. RMDM is capable of accommodating all of these variations on guided tours. However, for the purposes of this article, it suffices to consider a *generic* guided tour construct, shown in the middle of Figure 1.

The *grouping* construct is a menu-like mechanism that enables access to other parts of a hypertext document. A typical example of a grouping is the opening screen of many applications which serves the purpose of providing access to other indices, guided tours, etc. Indices are special kinds of groupings. We are currently investigating other useful grouping constructs, such as the multilevel hierarchical structures so common in knowledge classification schemes.

The conditions or *logic predicates* qualifying indices and guided tours determine which instances of an entity are accessible from the construct. For example, Figure 2-a shows a conditional guided tour of all associate professors. The predicate *Faculty(rank='associate')* indicates that only those entity instances of faculty whose rank attribute is *associate* participate in the guided tour. The right part of the figure shows an instance of such a guided tour. Figure 2-b is an example of a conditional index web. Here, access is granted via an index-like construct. There are also return links from each participating node to the index, as shown on the right of the figure. Lastly, conditional indexed guided tour webs combine indices and guided tours to provide a richer access structure.

We will use the Stern School’s Information Systems Department’s Handbook application (*ISweb*) as a basis for discussion (URL: <http://is-2.stern.nyu.edu/isweb>). V. Balasubramanian et al. present a different application, also developed with RMM, in their sidebar in this issue. The ISweb hand-

book contains descriptions of the graduate programs and courses offered by the IS Department and a list of faculty members in the department together with their research interests. We have chosen this application for various reasons: (1) the acquaintance of many readers with this domain; (2) its moderate complexity enables an illustration that is rich in details, and (3) it illustrates the problems that arise when updates are relatively frequent (semi-annual in this case).

Figure 3 shows the complete RMDM diagram for the handbook application. Note that, in contrast to an entity-relationship diagram that represents the design of a database, an RMDM diagram describes how users will navigate a hypermedia application. To avoid cluttering, slices are not included in Figure 3, and only the key attributes of entities are shown. At the top of Figure 3 the grouping mechanism implements a “main menu”. 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 back and forth among all faculty members (ordered alphabetically). From the *faculty* entity, there is a *conditional index* into *courses* with predicate  $teaches(F, C)$ . The reciprocal index  $taught\_by(C, F)$  can be accessed from *courses*. Together, these two indices represent a many-many relationship between *faculty* and *courses*.

INSERT FIGURE 3 ABOUT HERE

The “teaches” conditional index allows the user to move from the faculty entity to the courses taught by that faculty member. Had we provided an *indexed guided tour* for the courses taught by a faculty member, the user would be able to choose which course to visit first and from there, she could use the next and previous links to visit other courses taught by the same faculty member.

## The Relationship Management Design Methodology (RMM)

INSERT FIGURE 4 ABOUT HERE

The RMM methodology is shown graphically in Figure 4 within the context of the complete

software development cycle. RMM focuses on the design, development and construction phases. In this article we concentrate on the design of access mechanisms, which is achieved through the first three steps of the methodology (shown in the shaded area of Figure 4). Although feasibility, requirements analysis, and testing are undeniably important phases in software development, they are beyond the scope of this article. To evaluate the application, one can use techniques like those proposed by Garzotto, Paolini and Mainetti (in this issue) and by Schneiderman [3].

The labels on the arrows in Figure 4 represent the various intermediate artifacts generated through the use of the methodology. Although present in the methodology, we do not show the feedback loops among the remaining stages to avoid cluttering the figure. Feedback loops in between the RMM design stages are shown by dashed lines.

The RMDM data model provides a strong prescription for choosing the nodes and links in the hypermedia application. However, many design issues must be decided independently by the designer (see the article by Thüring, Hannenmann and Haake in this issue). While our main purpose here is to outline the design methodology, we also discuss some design guidelines for each step.

### **Step S1: E-R Design.**

The first design step is to represent the information domain of the application via an Entity-Relationship diagram (E-R). The E-R representation has been chosen because it is familiar to system analysts, is well-documented [6], and can model information dependencies in numerous application domains. This stage of the design process represents a study of the relevant entities and relationships of the application domain. These entities and relationships form the basis of the hypermedia application and many of them will show up in the final application as nodes and links in a hypermedia web. In many situations the E-R diagram might already be available, for example, if the target application is a hypermedia interface to an existing database. In this case it can be reused directly in this step.

INSERT FIGURE 5 ABOUT HERE

Figure 5 shows the E-R diagram for the handbook application. The entities *Faculty*, *Courses*, *Programs*, etc., are shown in rectangles. The relationships *Teaches*, *Taught\_by*, *Pre-Requisite*, etc. are shown by dashed lines. In RMDM, the relationships that appear in E-R diagrams are called *associative* relationships, because they represent associations between entity instances. The graphical notation also shows the arity of each relationship. Namely, when an arc opens up into three lines, the arity is *many* on that side of the relationship. The possible arities of relationships, in accordance with the entity-relationship framework are *one-one*, *one-many* and *many-many*. To facilitate design of navigation in stage S3, we use standard database design techniques to split *many-many* relationships into *one-many* ones. In Figure 5, for example, the two one-many relationships *Teaches* and *Taught\_by* originate in a many-many relationship between *Faculty* and *Courses*. During the navigational design step S3, relevant relationships are identified and are made available for navigation via indices or guided tours.

### **E-R Design Guidelines:**

Since E-R design has been extensively discussed elsewhere (see for example [4]), we will confine ourselves to but a few remarks here. The objective in the design of hypermedia applications is to make the links between objects explicit as these are the main paths via which the user browses the individual items of information. An analysis of the domain using the E-R approach helps identify important relationships across which navigation can be supported. Conversely, if a navigational path across entities is a requirement for the application, a corresponding associative relationship has to be included in the E-R design.

### **Step S2: Slice Design.**

This step, which is unique to hypermedia applications, determines how the information in the chosen entities will be presented to users, and how they may access it. It involves splitting an entity into meaningful slices and organizing these into a hypertext network. In its simplest form, all the information in an entity can be displayed within one window with scroll bars. Although such an approach is simple for the developer, it may be undesirable for a user, who may become disoriented when scrolling within large windows. Alternatively, the information can be divided into meaningful units that can be presented as separate but interrelated wholes. For example, Figure 6, shows the *Faculty* entity subdivided into four slices containing (1) general information, (2) a short biography, (3) research interests, and (4) a video-clip.



The organization of entities into slices is called the *slice design* phase, and it results in a *slice diagram*, as depicted in Figure 6. Each slice groups one or more attributes of the entity. For example, the *General* slice in Figure 6 groups the attributes: *first name*, *last name*, *general description* and *rank*, while the *Biography* slice contains *first name*, *last name* and *biography*. Each entity has a distinguished slice, its *head*, which is used as a default to anchor links coming into the entity. In the diagram, entity heads are marked with an asterisk.

The entity diagram also models navigation between slices via uni- and bi-directional links. The labels on bi-directional links name both directions. The up/left direction appears in parenthesis. For example, the link joining *General* and *Research Area* is labeled *expertise* in one direction (from *General* to *Research Area*), and *Faculty* in the other direction. These links, which represent connections between slices, are called *structural links*<sup>3</sup> to differentiate them from the associative relationships appearing in the E-R diagram. Structural links differ from associative relationships in that the former connect information pieces within the same entity instance, while associative relationships interconnect different entity instances belonging, in most cases, to different entity classes.

From a navigational point of view, there is an important reason for differentiating among these two kinds of connections. When a user traverses an *associative* link, the information context changes, for example, from a *faculty* to a *course*. However, when a structural link is traversed, the information context remains within the same entity. To support the implementation of different user-interface cues for navigation, structural and associative connections are differentiated graphically in RMM artifacts: structural connections are drawn with solid lines, whereas dashed lines are used for associative relationships (as in Figures 3 and 5). The concepts of structural and associative relationships also surface under different perspectives in other articles in this special issue. For example, they relate to *local* and *global* coherence in Thüring, Hannemann and Haake's article and to *local* and *global* navigation in Kahn's sidebar (in this issue). Together, the entities and relationships obtained in the E-R design step and the slice links comprise the structural components defining the *application schema* described in Bieber and Kacmar's paper (this issue). The output of the entity design phase is an enriched E-R diagram, denoted E-R<sup>+</sup>, which is obtained from the E-R diagram by *filling* each entity with its slice design diagram.

---

<sup>3</sup>This is consistent with HDM notation[9].

INSERT FIGURE 6 ABOUT HERE

### Slice Design Guidelines

There are four main considerations: (1) dividing an entity into slices, (2) choosing one slice to be the *head* of the entity, (3) interconnecting the various slices, and (4) labeling the links. Regarding the first issue, it is important to remember that each slice will represent a *whole* for the system user. Thus, slices should group only related information items but should not contain too much information. Ideally, scroll bars should not be necessary, because users tend to lose focus when using them.

Choosing the *head* slice and deciding on the interconnections between slices requires an analysis of the domain. In our case, the *General* slice has been chosen to be head of the *faculty* entity because it is the most representative of the slices. The links reflect the need to connect more general units, e.g., *general*, to more specific ones, e.g., *biography*, *research area* and *video-clip*, and to provide back links to navigate back from these. Finally, choosing appropriate link labels and anchors is a delicate issue as Thüring, Hannemann and Haake (in this issue) note. In our applications, we picked labels that explicitly highlight the nature of the target slices.

### Step S3: Navigational Design.

In this step, we design the paths that will enable hypertext navigation. Each associative relationship appearing in the E-R<sup>+</sup> diagram is analyzed. If, according to the requirements analysis, an associative relationship should be made accessible for navigation, it is replaced by one or more RMDM access structures. Since the RMM methodology is meant for domains that are updated on a relatively frequent basis, all navigational paths are specified in generic terms. This means that there are no hard coded links between instances of entities; instead, links are specified by referring to properties of entities and of relationships. The three RMDM navigational elements that make this possible are conditional indices, conditional guided tours and conditional indexed guided tours.

INSERT FIGURE 7 ABOUT HERE

One starts step S3 by designing the navigation between entities, which is based on associative relationships. For example, the *Teaches* relationship between faculty and courses is used to access all courses taught by a faculty member, and the *Taught\_by* relationship is used to access the information about all faculty members teaching a course. As shown in Figure 7-a we used conditional indexed guided tours in this case. The name of the relationship is used as a condition in these access structures to indicate which instances of the entities are to be interconnected. This ensures for example, that only the courses taught by faculty member *F* will appear when traversing the *Teaches* link from faculty *F*'s node.

Next, we design high level access structures by grouping items of interest. Figure 7-b illustrates how to use groupings to provide hierarchical menu-like access to courses and faculty in the IS handbook. These menus are an alternative to the design presented in Figure 3. The grouping shown at the top of Figure 7-b represents a menu with two choices: *faculty* and *courses*. Below this first menu and to its left is another grouping construct, that provides access to an alphabetical list of faculty (in the form of a guided tour) and to an index of faculty by rank.

By default, access structures enter an entity via its head slice. However, designers can specify a different entry point. This can be indicated, for example, by tagging the access structure with the name of the target slice. At the end of the navigational design stage, the ER<sup>+</sup> diagram has been transformed into an RMDM (Relationship Management Data model) diagram, like the one shown in Figure 3, which describes all access structures to be implemented in the system.

### **Navigational Design Guidelines**

The design can proceed *bottom-up*, by focusing first on each entity and then on the more general access structures, or the inverse process can be adopted, resulting in a *top down* approach. The approach discussed here has been bottom-up, mainly, because we followed a very structured design process. However, we anticipate that software developers will want to approach the design process in a bottom-up, or middle-out fashion. As Nanard and Nanard point out in their paper, these requirements are an important consideration in the design of an RMM-based computerized environment for hypermedia design and development [5].

During stage S3, designers have to identify (a) the information components and relationships that will be accessed, (b) what groupings are to be present and (c) what access structures to use in

each case. Decisions regarding (a) and (b) must be based on system requirements obtained during the requirements analysis phase, reflecting the characteristics and logical structure of the application domain. Further refinements through user participation should be encouraged during the S3 design step. Regarding (c), we use a few simple guidelines. *Groupings* provide hierarchical points of entry; whenever possible, we try to reduce the depth of hierarchies to avoid user disorientation [17]. Alternatively, graphical cues can be used to provide an overview of where a reader is in the hypertext network. One-one relationships are implemented with bidirectional links. For one-many relationships the choice is more complex: between a *guided tour*, an *index* and an *indexed guided tour*.

We prefer a *guided tour* to an *index* when the number of participating entity instances is relatively small (say less than ten) and when there are no index keys that can be of help to users. For example, the research areas of a faculty member are organized as a guided tour because, for most students using the system, the names of the research areas are not very informative. On the other hand, when there is a large number of instances in the presence of a meaningful key, an index is better suited. Indexed guided tours are a hybrid; often used when there is a suitable index key and some local navigation is desired, as is the case with the courses taught by a professor, where the course number serves as a key.

## User-Interface Design and Construction

We now describe the remaining four steps of the RMM methodology. Because they do not deal directly with the design of access mechanisms, we only provide a brief discussion in this article.

Most current hypermedia building kits such as Toolbook, Hypercard, Macromind Director, as well as tools used to create HTML documents, offer some degree of support for software development. For example, Toolbook and HyperCard provide graphical widgets for building code, and libraries containing hypermedia programming constructs. However, these tools only provide building blocks that facilitate the programming stage of software development, without addressing the broader design and development issues that have been outlined above. Step S4, *conversion protocol design*, uses a set of *conversion rules* to transform each element of the RMDM diagram into an object in the target platform. For example, a Toolbook list-box or an HTML form can be used to implement an index. Step S4 is currently performed manually by programmers. However, a group at New York University is developing an RMDM to HTML compiler.

Step S5, *user-interface design* involves the design of screen layouts for every object appearing in the RMDM diagram obtained in step S3. This includes button layouts, appearance of nodes and indices and location of navigational aids. Decisions about how link traversal, history, backtracking and navigational mechanisms are to be implemented, are taken during step S6, *run-time behavior design*. Also during this stage, developers consider the volatility and the size of the domain to decide whether node contents and link endpoints are to be built during application development or dynamically computed on demand at runtime (see V. Balasubramanian et al's sidebar in this issue). The RMM methodology, although geared towards the latter, also supports the former. Finally, step S7 consists of *construction and testing*, as in traditional software engineering projects. In hypermedia applications, special care needs to be taken to thoroughly test all navigational paths.

INSERT FIGURE 8 ABOUT HERE

A sample application developed using RMM is shown in Figure 8, which depicts a screen from the WWW implementation of the IS Handbook. The figure shows the head slice of an instance of the *Courses* entity, containing information about the course C20.0001, the undergraduate introductory course in information systems. The map appearing at the top right hand side of the page is obtained from the RMDM diagram. This drawing is a click-able map that can be used as an alternative means of navigation, in the spirit of Intermedia's Web views [25], and other graphic-based navigational approaches [19]. Such graphs can be constructed following spatial design principles as discussed by Marshall and Shipman (in this issue), and are available as a by-product of the RMM methodological process. We are currently designing a suite of software tools to support the design and development of RMM-based applications [5], which we expect to utilize to further evaluate the RMM methodology.

## Conclusion and Future Work

As demand for hypermedia products increases, there is a need to replace the current *ad-hoc* design and construction approaches, which are highly labor intensive and costly, with more efficient approaches that provide guidelines for project managers, produce standardized documentation and

give automated support for developers. Our objective in this paper was to present a hypermedia design and construction methodology that addresses these issues. The proposed RMM methodology is most suited to applications that have a regular structure, especially where there is a frequent need to update the information to keep the system current. Many commercial applications, including product catalogs, electronic commerce gateways, design manuals and interfaces to database management systems fit this description. We believe that the RMM methodology can serve as the basis for the design and development of robust hypermedia applications.

## Acknowledgments

This research was partially supported by a hardware grant from Apple Inc. We wish to thank the reviewers for their insightful comments; the following people for their help in implementing the department handbook: Terrance Chen, Hon-Fun Kun, Alex Bronstein and Rama Krishnan, and Ravi N. Arunkundram for his assistance in preparing this manuscript.

## References

- [1] 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.
- [2] P. Balasubramanian, Tomas Isakowitz, and Edward. A. Stohr. Designing hypermedia applications. In Ralph H. Sprague and Bruce Shriver, editors, *Proceedings of the 27th Hawaii International Conference on Information Systems (HICSS)*, pages 354–365, Maui, HI, January 1994. IEEE Computer Society Press.
- [3] Rodrigo Botafogo, Ehud Rivlin, and Ben Schneiderman. Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics. *ACM Transactions on Office Information Systems*, 2(10):142–180, April 1992.
- [4] Peter P. S. Chen. Database Design Based on Entity and Relationship. In S. Bing Yao, editor, *Principles of Database Design, Volume 1: Logical Organizations*, chapter 5, pages 174–210. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, USA, 1985.

- [5] Alicia Díaz and Tomás Isakowitz. RMCASE: Computer-Aided Support for Hypermedia Design and Development. Working paper IS-95-3, Center for Research in Information Systems, New York University, Information Systems Department, New York, NY 10012, 1995.
- [6] Ramez Elmasri and Shamkant Navate. *Fundamental of Database Systems*. The Benjamin/Cummings Publishing Company, second edition, 1990.
- [7] 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.
- [8] Franca Garzotto, Paolo Paolini, and Luca Mainetti. Navigation Patterns in Hypermedia Data Bases. In *Proceedings of the 26<sup>th</sup> Hawaii International Conference on System Sciences, Vol. III*, pages 370–379. IEEE Computer Society Press, January 1993.
- [9] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. HDM - A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Office Information Systems*, 11(1):1–26, January 1993.
- [10] J. Gaulding and B. Katz. Hypermedia Application Design: A Structured Approach. In Wolfgang Schuler, Joerg Hannemann, and Norbert Streitz, editors, *Designing User Interfaces for Hypermedia*, pages 43–52. Springer-Verlag, ESPRIT series, Heidelberg, December 1994.
- [11] Kaj Gronbaek and Randal H. Trigg. Design Issues for a Dexter-Based Hypermedia System. *Communications of the ACM*, 37(2):40–49, February 1994.
- [12] Frank G. Halasz and Meyer Schwartz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, February 1994.
- [13] Lynda Hardman, Dick C. C. Bulterman, and Guido van Rossum. The Amsterdam Hypermedia Model. *Communications of the ACM*, 37(2):50–63, February 1994.
- [14] Tomás Isakowitz and Manfred Thüring. (Eds.) Methodologies for Designing and Developing Hypermedia Applications. Working paper IS-94-8, Center for Research in Information Systems, Information Systems Department, Stern School of Business, New York University, New York, NY 10012, December 1994.

- [15] Danny B Lange. Object-Oriented Hypermodeling of Hypermedia Supported Information Systems. *Proceedings of the 26th Hawaii International Conference on System Sciences*, III:380–389, January 5-8 1993.
- [16] John J. Leggett and John L. Schnase. Viewing Dexter with Open Eyes. *Communications of the ACM*, 37(2):76–86, February 1994.
- [17] Jakob Nielsen. Through Hypertext. *Communications of the ACM*, 33(3):297–310, March 1990.
- [18] Amy Pearl. Sun's Link Service, A Protocol for Linking. In *Hypertext'89 Proceedings*, pages 137–146. ACM Press, November 1989.
- [19] Ehud Rivlin, Rodrigo Botafogo, and Ben Schneiderman. Navigating Hyperspace: Designing a Structure-Based Toolbox. *Communications of the ACM*, 37(2):87–96, February 1994.
- [20] 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.
- [21] Wolfgang Schuler, Joerg Hannemann, and Norbert Streitz, editors. *Designing User Interfaces for Hypermedia*. ESPRIT series. Springer-Verlag, Heidelberg, December 1994.
- [22] Daniel Schwabe and Gustavo Rossi. Building Hypermedia Applications as Navigational Views of Information Models. *Proceedings of the 28th Hawaii International Conference on System Sciences*, III:231–240, January 1995.
- [23] P. David Stotts and Richard Furuta. Programmable Browsing Semantics in Trellis. In *HyperText-89 Proceedings*, pages 27–42, 1989.
- [24] Frank W. Tompa. A Data Model for Flexible Hypertext Database Systems. *ACM Transactions on Information Systems*, 7(1), January 1989.
- [25] Kenneth Utting and Nicole Yankelovich. Context and Orientation in Hypermedia Networks. *ACM Transactions on Information Systems*, 7(1):58–84, January 1989.



## Figures

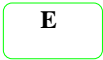


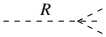


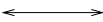

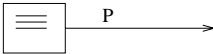
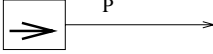
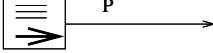
|                                     |  |
|-------------------------------------|--|
| <p><i>E-R Domain Primitives</i></p> | <p>Entities </p> <p>Attributes </p> <p>One-One Associative Relationship </p> <p>One-Many Associative Relationship </p>   |
| <p><i>RMD Domain Primitives</i></p> | <p>Slices </p>   |
| <p><i>Access Primitives</i></p>     | <p>Uni-Directional Link </p> <p>Bi-Directional Link </p> <p>Grouping </p> <p>Conditional Index </p> <p>Conditional Guided Tour </p> <p>Conditional Indexed Guided tour </p> |

Figure 1: The Relationship Management Data Model (RMDM) primitives. *E-R primitives* model how information is structured in the application domain. The *slice* domain primitive models how information is to be presented. The *access primitives* model navigation.

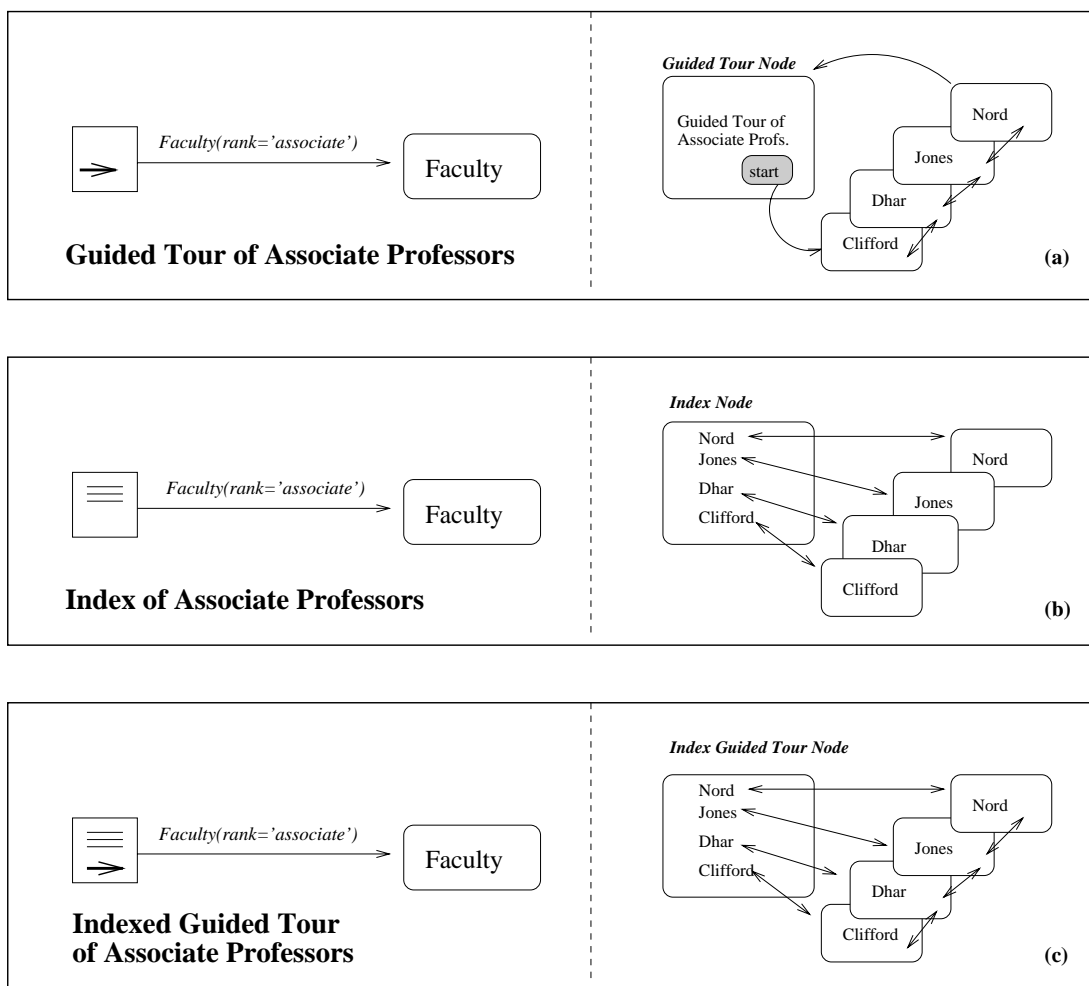


Figure 2: Examples of the conditional RMDM access constructs.

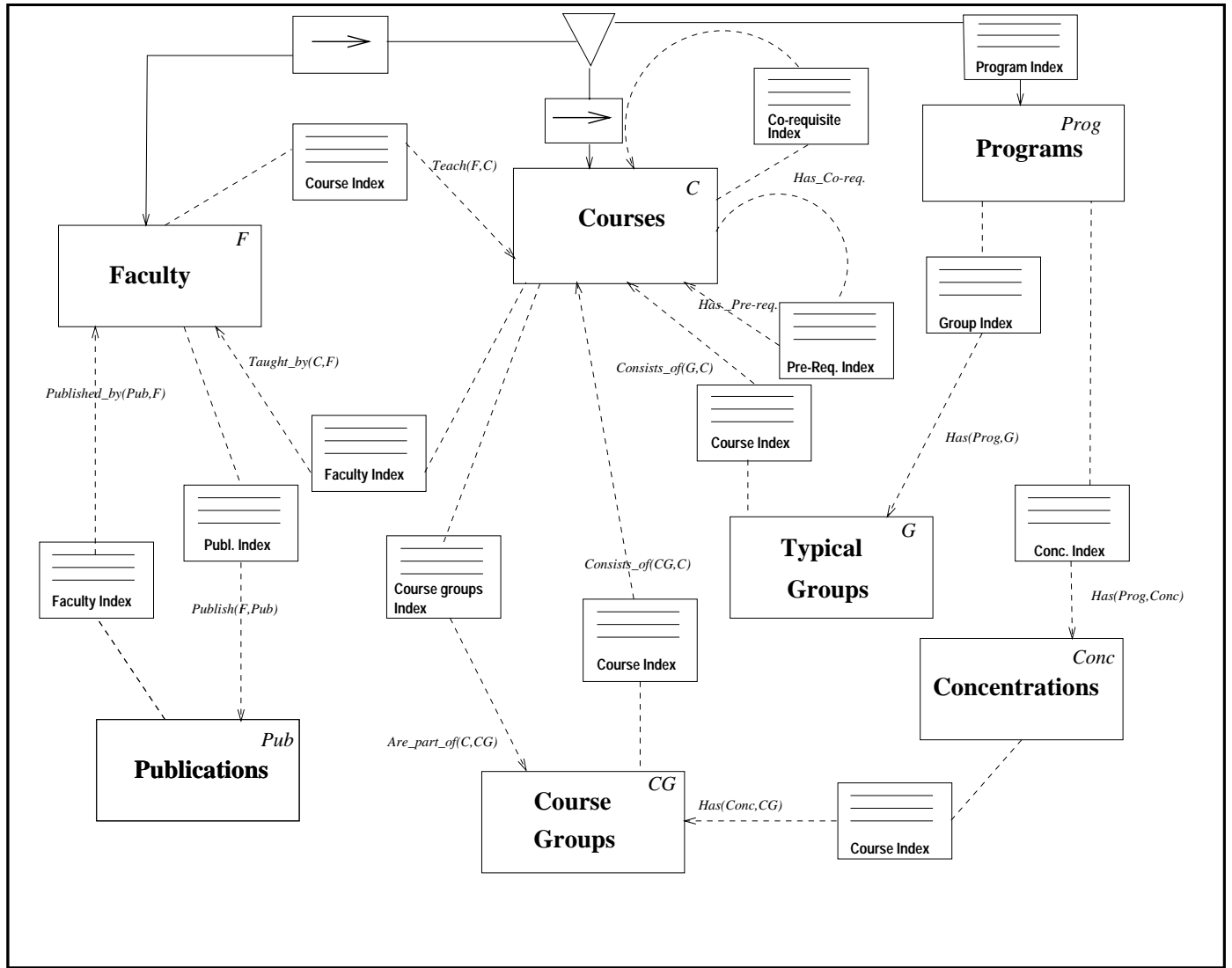


Figure 3: RMDM Diagram for the Information Systems Department Handbook application represents the ultimate outcome of the three steps of the design process.

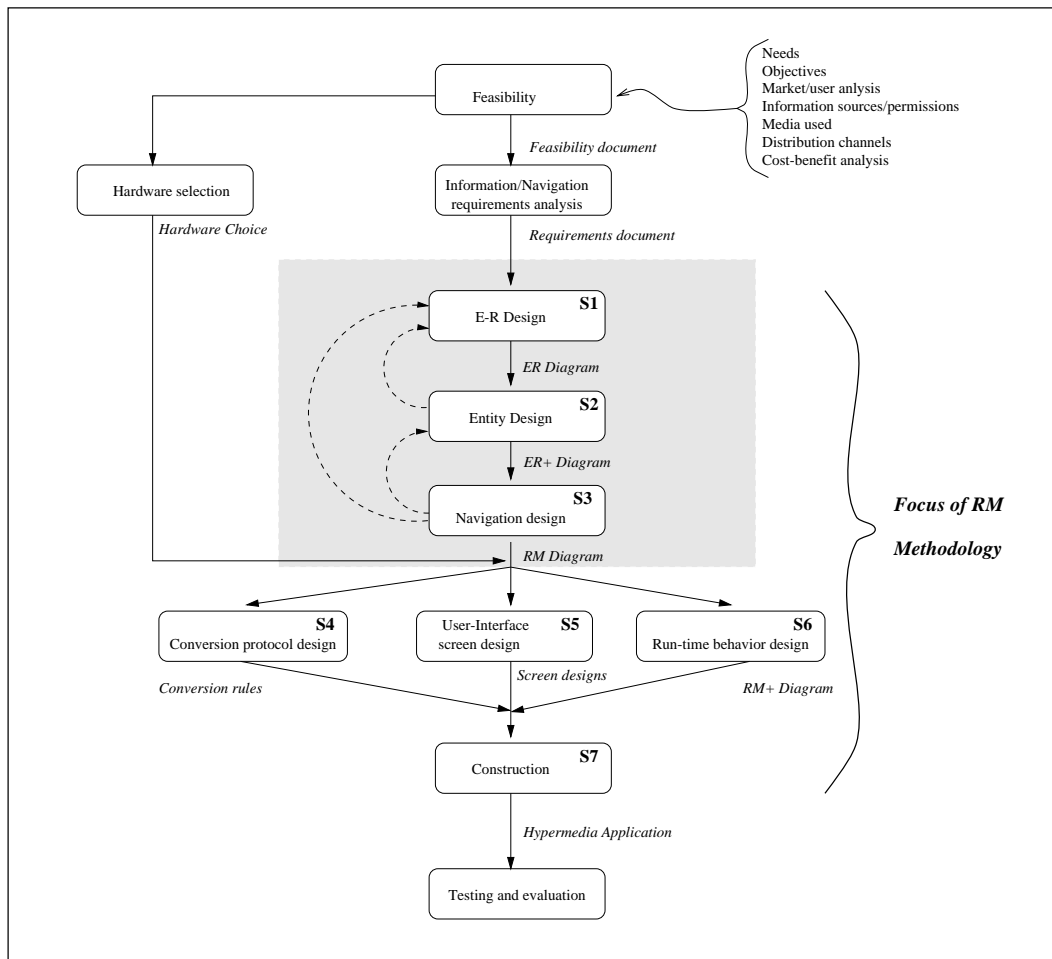


Figure 4: The RMM Design Methodology. The arrows connecting the various stages are tagged with the objects to be used as input for the next step. Our focus in this article is on the design steps, shown in the shaded area. To avoid cluttering, we only show feedback loops within the design phase (the dashed lines). The remaining feedback loops, although present in RMM, are not shown.

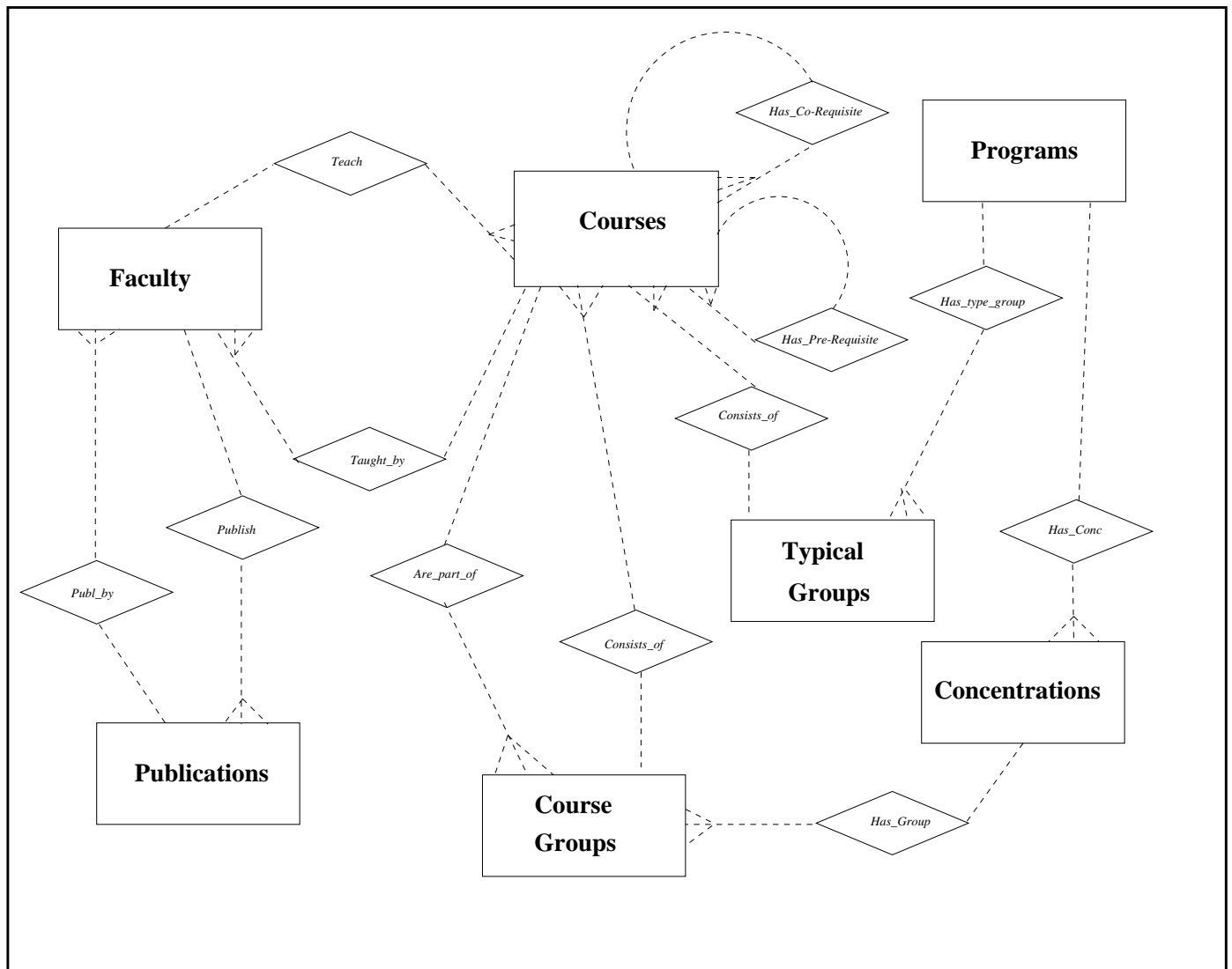


Figure 5: E-R Diagram for the handbook application. Entities appear within rectangles and relationships as dashed lines. Relationship names appear within diamonds.

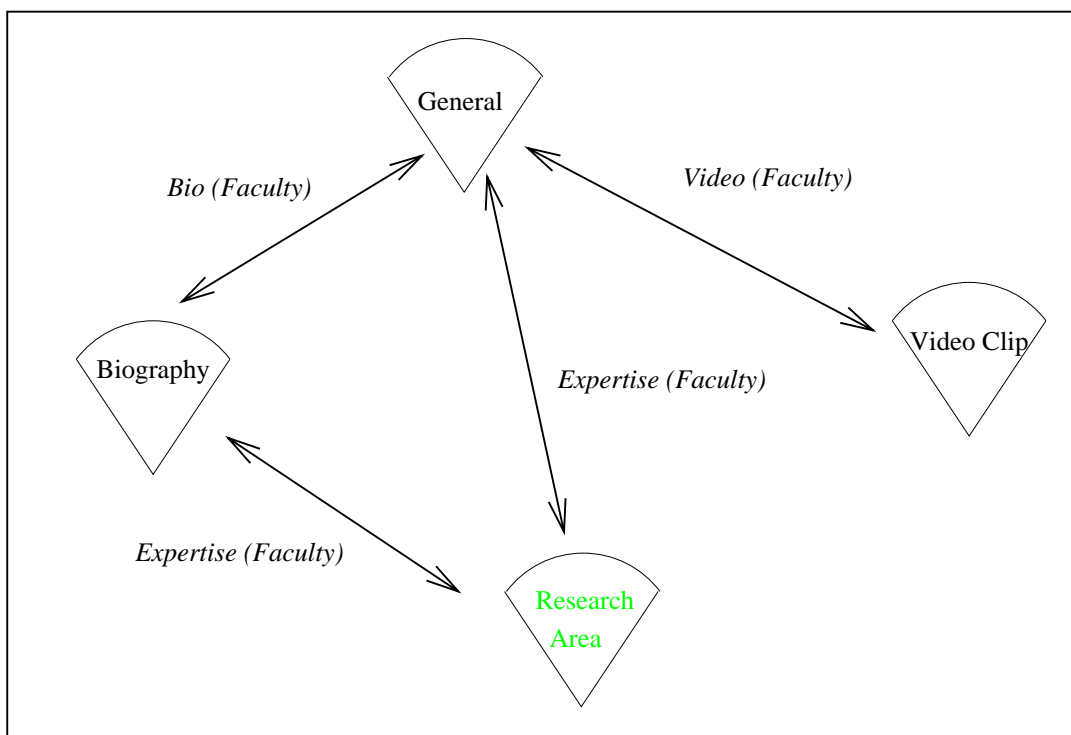


Figure 6: Slice diagram for faculty

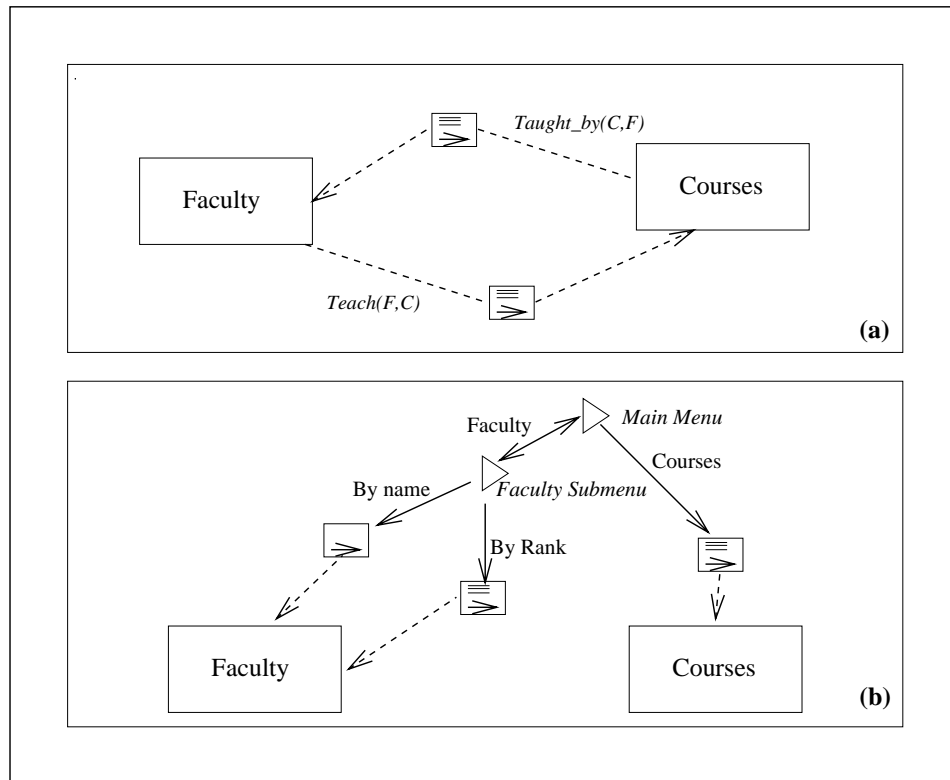


Figure 7: Examples of a Relationship Management Design Diagram. In (a), Conditional indexed guided tours implement the associative one-many relationships *Teach* and *Taught\_by*. The use of conditional access structures to provide access beyond that derived from associative relationships is illustrated in (b). Also note the two groupings, which implement a hierarchical menu-like access mechanism.

The screenshot shows a web browser window with the following elements:

- Browser Menu:** File, Edit, View, Go, Bookmarks, Options, Directory, Help.
- Navigation Buttons:** Back, Forward, Home, Reload, Images, Open, Print, Find, Stop.
- Location Bar:** http://is-2.stern.nyu.edu/isweb/courses/C
- Navigation Links:** What's New, What's Cool, Handbook, Net Search, Net Directory, Newsgroups.
- Course Title:** Computer Based Systems for Management Support C20.0001.
- Course Details:** Available in: Fall, Spring, Summer 1; Points: 3; Type: Introductory.
- Click-able Map:** A diagram showing relationships between Courses, Faculty, Programs, and Taught by. Courses are linked to Faculty (Teaches) and Programs. Faculty are linked to Courses (Taught by) and Programs. Courses also have links to Description and Syllabus.
- Description:** This introductory course trains students to (a) identify problems and opportunities that can benefit from information systems support; (b) use structured modeling techniques to evaluate, design, and specify simple information systems; (c) apply state of the art software tools to implement such systems in practice; and (d) appreciate the crucial role that information technology will play in business and in society in the 2000's.
- Text:** Throughout the course, the students gain considerable experience in using software packages in the areas of spreadsheet modeling, database management, and visual programming. Students who have previous knowledge of information systems and computer science material can try to substitute C20.0001 with C20.0010 -- an advanced version of the former. To qualify for this substitution, the student must pass a proficiency examination (administered by the Stern Undergraduate Dean's office).
- Navigation:** Taught by and Syllabi links with arrow icons. Buttons for 'Click here to go to the Courses Index' and 'Click here to Return to the IS Dept Home Page'.

Figure 8: A course page from the WWW implementation of the IS handbook. The click-able map on the upper right corner is a by by-product of the RMM methodology. It serves to orient users, and is also an alternative means of navigation.