The Extended RMM Methodology for Web Publishing

Tomas Isakowitz
University of Pennsylvania

Arnold Kamis
New York University

Marios Koufaris
New York University

May 1998

Working Paper Series
Stern #IS-98-18

The Extended RMM Methodology for Web Publishing♣

TOMAS ISAKOWITZ[1] **ARNOLD KAMIS** **MARIOS KOUFARIS**

University of Pennsylvania **New York University** **New York University**

The Relationship Management Methodology (RMM) for hypermedia design was originally introduced in 1995, and has since evolved in a number of ways in response to the rapid growth in demand for hypermedia applications on the World Wide Web. The revamped methodology is demonstrated in the design of rich web applications. Design and implementation issues are discussed, including database integration, and top-down versus bottom up approaches to Web Information System (WIS) application development. The graphical and programming language notations for RMM's new constructs are presented. RMM promotes sound design and maintainable development of hypermedia.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design—*data models;* H.2.3 [**Database Management**]: Languages—*Query languages;* H.2.4 [**Database Management**]: Systems—*Relational databases;* H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*Architectures;* H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*Navigation*

General Terms: Design

Additional Key Words and Phrases: data model, entity-relationship diagram, hypermedia, hypertext, information domain, maintainability, reuse, structured design, web-based information system

♣ This research was supported by grants from NTT America and Sun MicroSystems.

[1] Authors' addresses: T. Isakowitz, Wharton School of Business, University of Pennsylvania, 3620 Locust Walk, Phila. PA 19104 USA; email: ti@wharton.upenn.edu; A. Kamis, MEC 9-181, New York University, 44 W. 4th St., NY, NY 10012 USA; email: akamis@stern.nyu.edu; M. Koufaris, MEC 9-181, New York University, 44 W. 4th St., NY, NY 10012 USA; email: mkoufari@stern.nyu.edu.

1. Introduction

Tim Horgan, Chief Technology Officer of CIO Communications, Inc., synthesized findings from case studies of large intranets at Bay Networks, Booz-Allen Hamilton, Digital, Ford, Sun, and US West.[2] He found that a central information architecture and shared, common mechanisms/services were helpful for coping with problems of scalability and "information anarchy". As Nielsen points out, "On the Web, the only constant is change, and it is important to design a scalable information architecture that can accommodate such change. A site that works perfectly as long as it stays the same will quickly die" [Nielsen 98]. Horgan also noted two insights relevant to the importance of this paper: 1) hidden costs lurk in larger, complex intranets, and 2) good information designers are hard to find.

1.1 Design of Web-Based Information Systems

The web platform has transformed itself in the few years since its inception in 1993 from an instrument used merely to establish an on-line presence to a platform that can support all facets of organizational work. As a result, more important IS efforts are geared towards the development of information systems based on Web technology, denoted "Web-based Information Systems (WISs). [Isakowitz and Bieber 1998].

WIS application development has been approached from two main directions. There are graphic designers who focus, for the most part, on the aesthetics and marketing aspects, and there are programmers who approach WIS design as the construction of interconnected Perl or Java programs. What is lacking in these approaches is an information-design approach, one that leverages the structure of the information domain to drive the design of the WIS. Such an approach to design is relevant to information systems professionals, as well as to graphic designers and programmers. Our purpose is to introduce such an information-focused approach to WIS design, which complements, and is to be used in combination with, the graphical and the programmatic design approaches.

The Relationship Management Methodology (RMM) provides a structured design methodology for the development of a large class of hypermedia applications, facilitating their design, development and maintenance. This large class consists of moderate to high complexity applications containing reusable components. RMM is currently in use at financial institutions (e.g., Merrill Lynch), publishing houses (e.g., M.E. Sharpe, Inc.), research institutions (e.g., Bellcore) and educational institutions (Pace University in NY; SYRECOS consortium in Luxembourg, Staffordshire University in the UK).

After some initial experimentation, it became clear that the original RMM, as presented in 1995, suffered from limitations. Through a series of enhancements sparked by feedback from diverse projects and users, RMM has been adapted to the current publishing needs of Web-based Information Systems (WIS) The interaction aspects (data input and database updates) of WIS will be tackled in forthcoming research. This article presents the complete, revised publishing methodology, which retains its original name: RMM. Previous knowledge of RMM is not required as background for this article. See section 3 for a brief discussion of the limitations of the original RMM and how these are overcome by the solutions we present here.

1.2 Defining the Problem

In order to explain RMM's contribution, it is useful to differentiate three levels of modeling a WIS application, as illustrated in Figure 1. At the top, the **presentation level** deals with *how* the information is presented. At this level, one chooses information to group together, or separate and hyperlink, into presentation units, such as web pages. The **storage level** describes how information is organized physically, in terms of what software applications (e.g., databases, graphic editors) are needed, what files are used, in what kind of directory organization, etc. In between the storage and presentation

[2] "Developing Your Intranet Strategy and Plan", http://www.cio.com/WebMaster/strategy/tsld001.htm, 1998

levels, the **logical level** maps the information stored in databases and web servers onto the actual web pages seen by the user. This can be done through various modeling techniques, such as object-oriented design and E-R design.

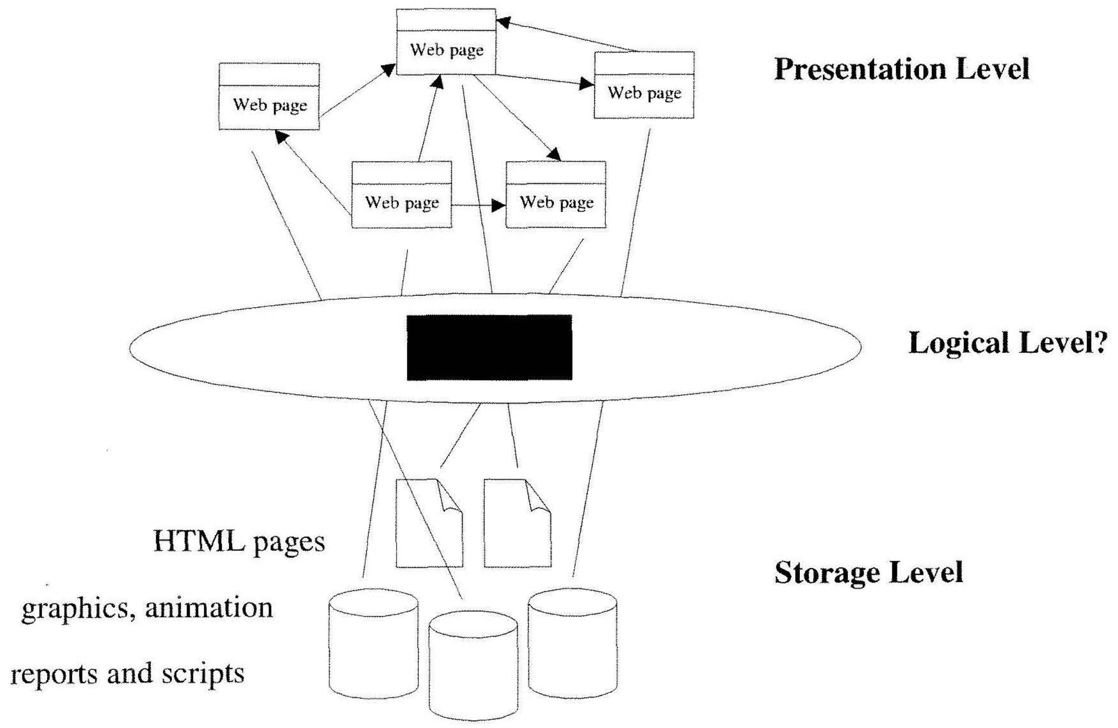


Figure 1: The three modeling levels of Web Information Systems

Graphic designers usually work at the presentation level where the central issues are the screens themselves, whereas systems developers are more interested in the physical storage layer, where they code scripts and construct efficient storage and retrieval mechanisms. Modeling techniques and tools have already been developed to support design activities at these levels. For example, HTML editors are used for the presentation layer and web-analysis tools for the storage layer. There are, however, no definitive modeling mechanisms to handle the logical level. Hence, currently, designers of web applications lack a terminology to refer to the logical level and a structured approach to working with it. This is where the main contribution of RMM lies; RMM proposes a modeling language to represent the information domain and navigational structure of WIS at the *logical* level. Based on its data modeling language, RMM also provides a methodology that drives design and development, and establishes effective bridges between the presentation and storage layers.

The main elements that RMM provides for logical modeling are identified in Figure 2 and explained thoroughly in the following sections. The m-slice and the application diagram are the main contributions defined in the new RMM. RMM's foundation is the E-R diagram, which is well established in the modeling of relational domains.

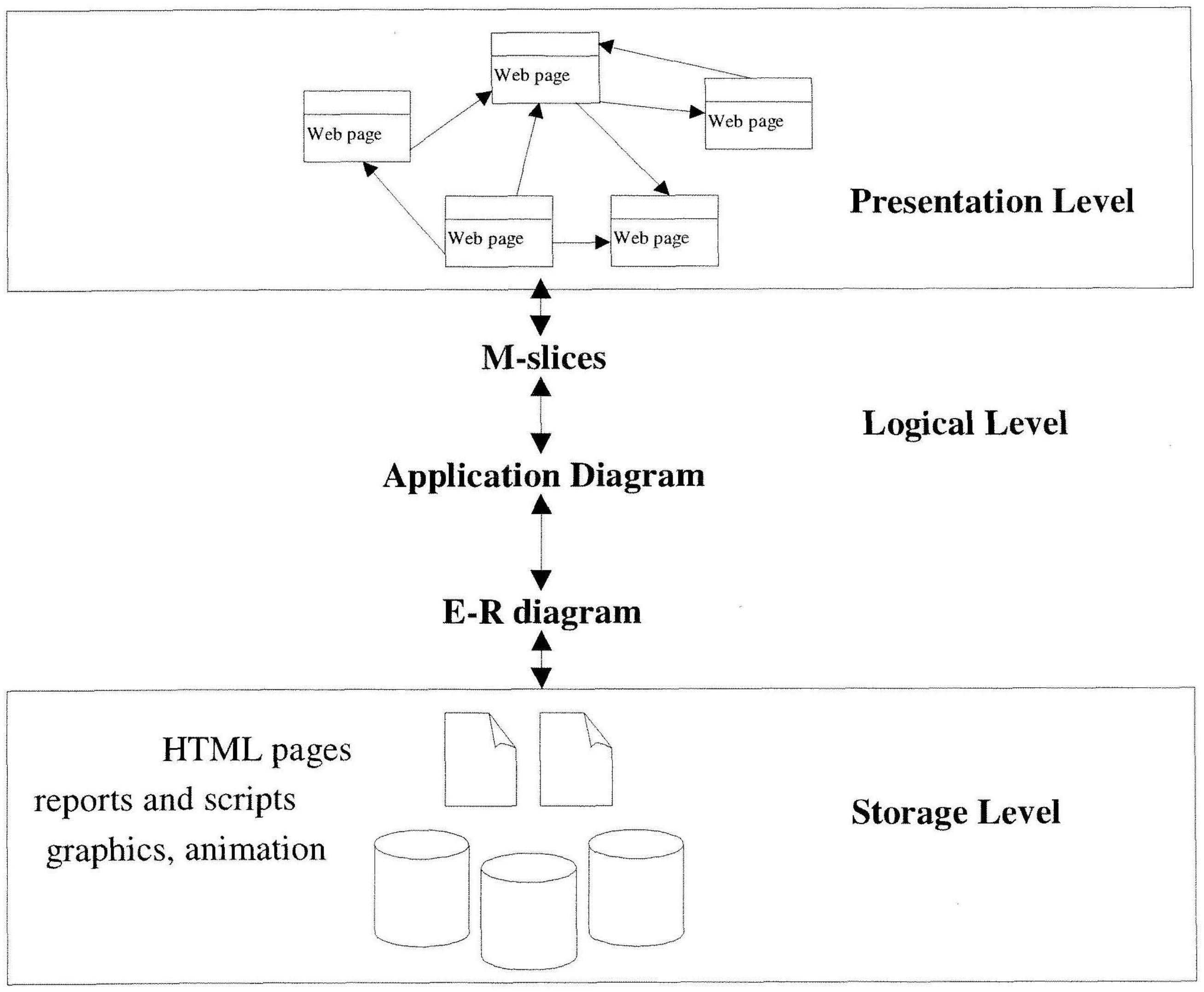


Figure 2: RMM's contribution in terms of the three layers of WIS modeling.

Conceptual integration among the three levels is needed to help determine, for example, how stored information is mapped through the logical level onto the presentation level. Typically, these mappings occur at a cognitive level only in developers' minds. A key contribution of RMM is the formalization of these mappings, which leads to the development of well-structured and easily maintainable WIS. As a result, software tools can be constructed to support RMM design, a key pre-requisite for efficient web development.

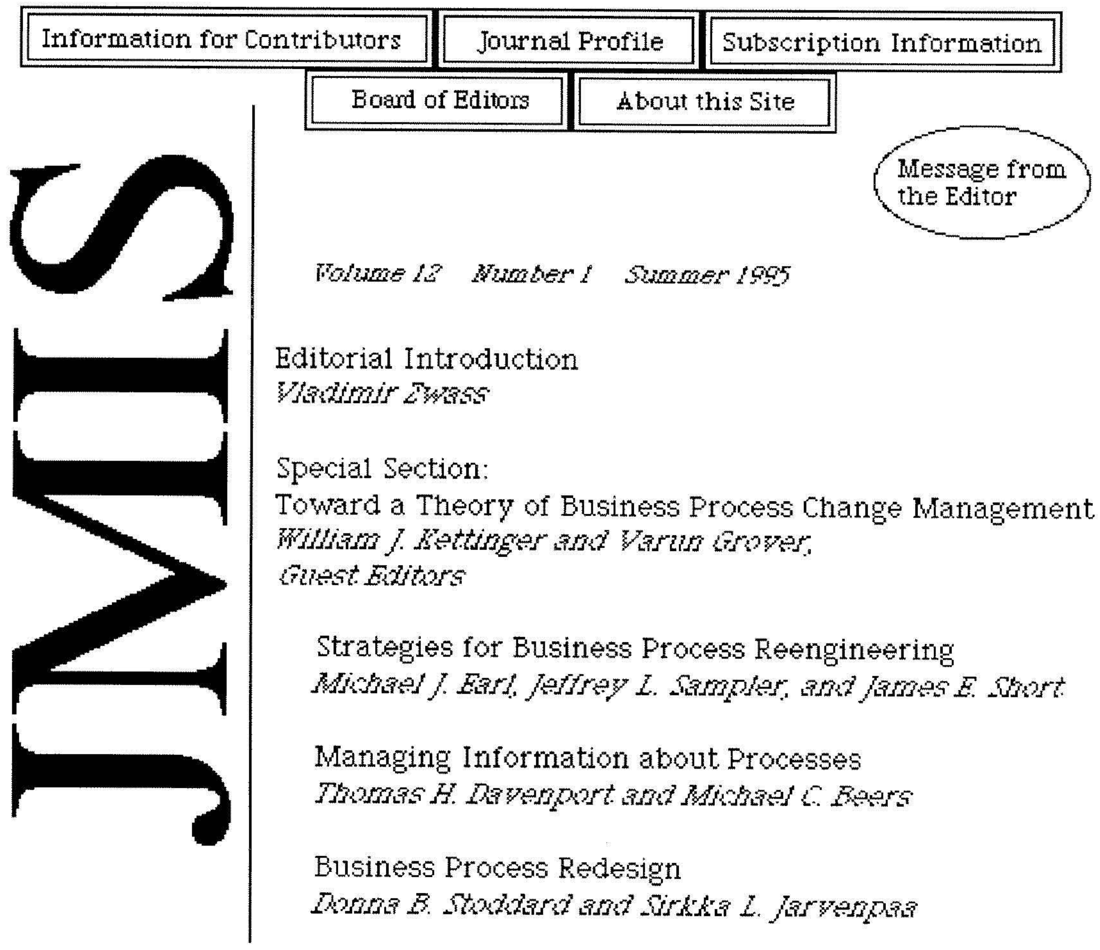
1.3 Organization of this Article

We first describe the web site for the Journal of Management Information Systems (JMIS), which we designed and developed using RMM, and which will be used to illustrate the various RMM modeling elements. Next we discuss the limitations of the original RMM and discuss briefly how the new extensions overcome these limitations. The next three sections describe RMM extensions that address these limitations, the revised RMM design process, which combines top-down and bottom-up approaches, and a section on implementing RMM applications with databases. We end with short *related research* and *conclusion* sections followed by two appendices. The appendices illustrate RMM's new data model in the graphical notation and in the programming language notation by way of a cumulative example.

2. The JMIS Web Site

The web site for the Journal of Management Information Systems (http://www.stern.nyu.edu/jmis) is moderately complex, and it contains reusable components. It is therefore a good example for illustrating the concepts in this paper. As shown in Figure 3, each issue has its own page with an index of all the articles and authors (a table of contents) featured in that issue as well as a number of buttons at the top of the screen providing information about the journal in general. We also constructed a top-level page, called the journal **toppage**, which lists all the issues of the journal currently available on the web site. There is a separate page for each article in which the abstract, its keywords and the authors' names are provided. Each author's name is hyperlinked to a page containing information about her or him. There is also a hyperlink to a keyword index that lists the available JMIS articles classified by each keyword. Throughout the site, references such as article titles, author names and issue volume-number-season, are hyperlinked to the appropriate pages.

The RMM diagram of the JMIS web site, as designed with the original methodology, is shown in Figure 4. One typically starts navigating at the site's homepage, "JMIS Homepage", which is a grouping in the original RMM. From there, one can click on the keyword index to obtain a list of all keywords. Clicking on one of them shows all the articles classified by it. One can then select an article and click on its title to show the article page. From there, one can click on a name to obtain information about an author.



Volume 12 Number 1 Summer 1995

Editorial Introduction
Vladimir Zwass

Special Section:
Toward a Theory of Business Process Change Management
William J. Kettinger and Varun Grover, Guest Editors

Strategies for Business Process Reengineering
Michael J. Earl, Jeffrey L. Sampler, and James E. Short

Managing Information about Processes
Thomas H. Davenport and Michael C. Beers

Business Process Redesign
Donna B. Stoddard and Sirkka L. Jarvenpaa

Figure 3: The issue table of contents page.

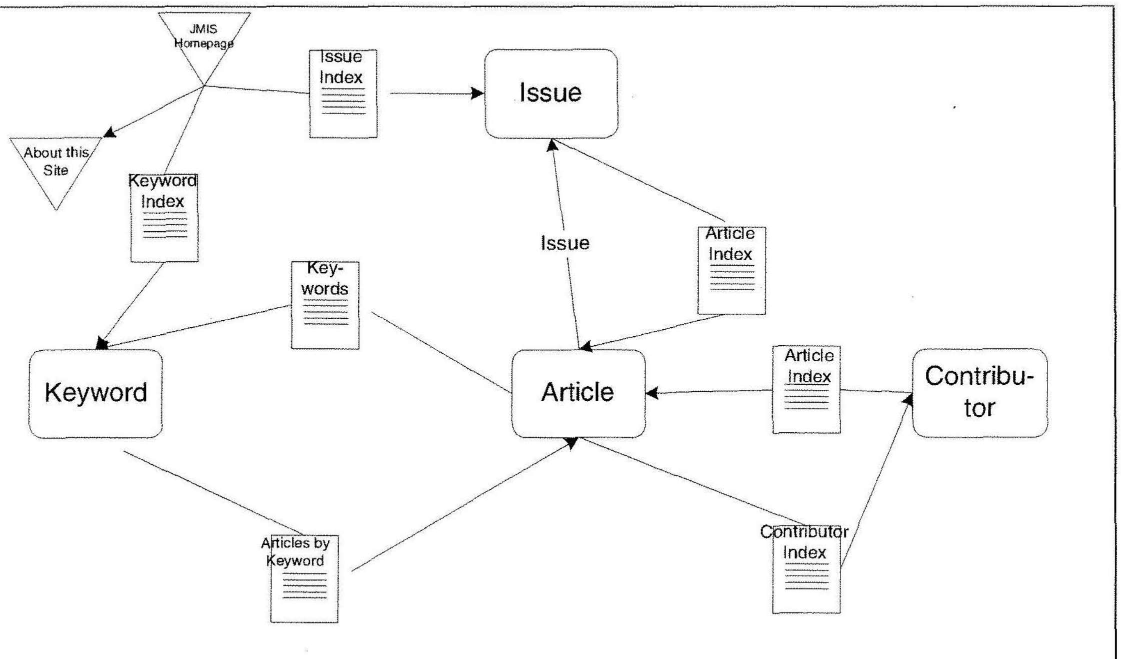


Figure 4: The RMM diagram for the JMIS web site.

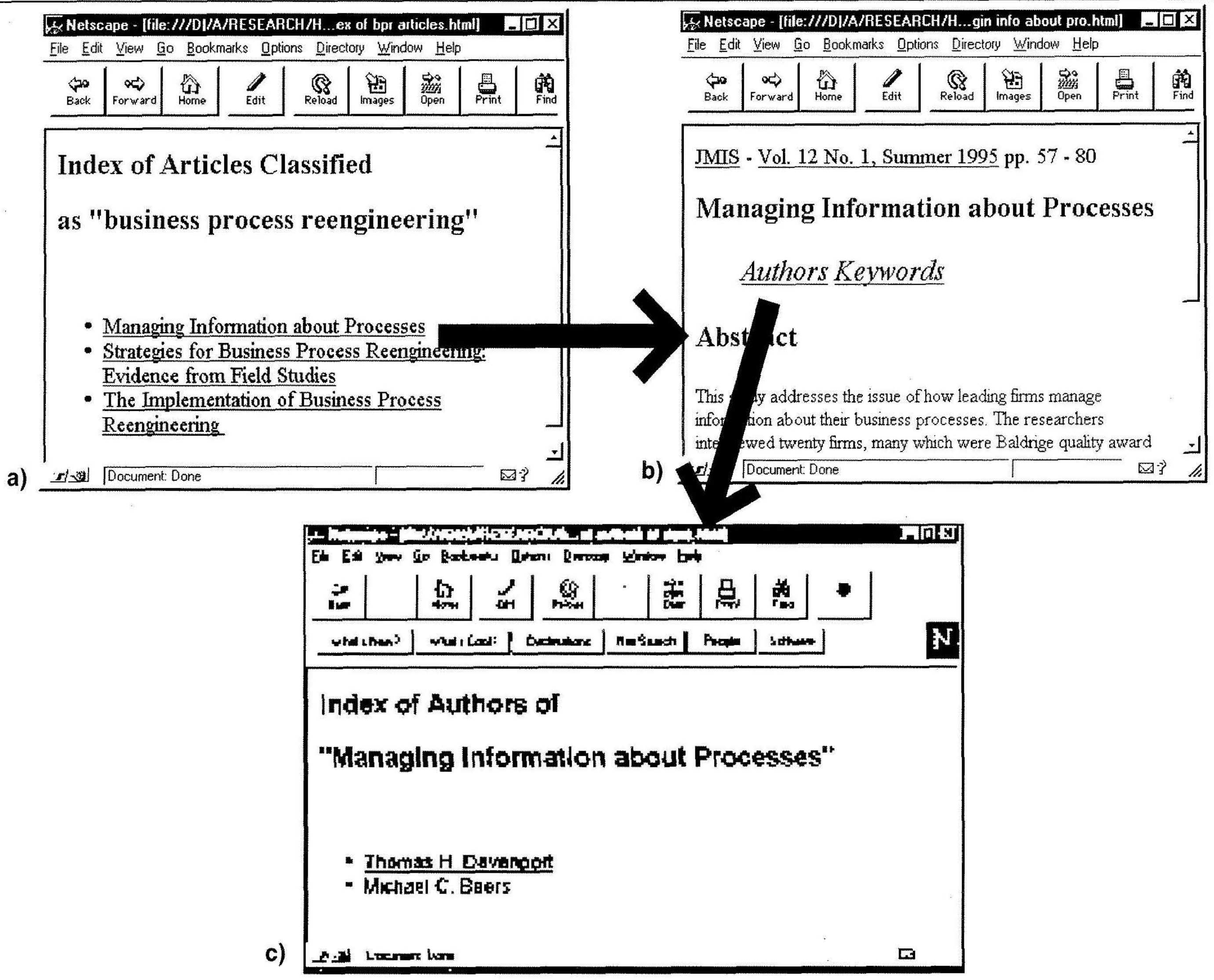
3. Limitations of Original RMM

This section describes some of the limitations of the original RMM methodology, and may be skipped by those readers only interested in the newest developments. It is provided here for the benefit of the readers interested in the development of the methodology and its data model.

The original RMM did not account for issues beyond the basic navigational structure of a hypermedia application. Perhaps the most serious limitation was RMM's inability to model contents of complex web pages. It was not possible, for example, to combine information from different entities in a single screen. Another serious limitation of RMM lay in that it forced a top-down software development process. Not only did we find that this constrained software development, it also discouraged re-use of low-level designs. Reusing high level designs was and is uncommon at this stage of WIS.

3.1 Poor Slice Contents (Limitation 1)

To appreciate the restrictiveness of these limitations, consider an original RMM implementation of the JMIS web site. Suppose we are looking for articles on Business Process Reengineering. Through the top page of the JMIS web site, we reach the index of all keywords and Business Process Reengineering, arriving at the page shown at the top left of Figure 5. Ideally, for each keyword one would like to see the list of full bibliographic references of all the articles classified by it. Instead, the pure RMM implementation of the JMIS web site, based on the diagram in Figure 4, results in a number of separate pages, as shown in Figure 5. Each page consists of either (a) an access structure (e.g., a grouping, index or guided tour), or (b) a page containing information elements of only one entity. The inability to aggregate, for example, the article title and the authors' names in the "Articles by Keyword" index (Figure 5-a), results in a cumbersome implementation.


a)
Netscape - [file:///D|/A/RESEARCH/H...ex of bpr articles.html]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find
Document: Done
b)
Netscape - [file:///D|/A/RESEARCH/H...gin info about pro.html]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find
Document: Done
c)
Document: Done

Index of Articles Classified

as "business process reengineering"

- Managing Information about Processes
- Strategies for Business Process Reengineering: Evidence from Field Studies
- The Implementation of Business Process Reengineering

JMIS - Vol. 12 No. 1, Summer 1995 pp. 57 - 80

Managing Information about Processes

Authors Keywords

Abst[illegible]ct

This [illegible]y addresses the issue of how leading firms manage info[illegible]tion about their business processes. The researchers inte[illegible]ewed twenty firms, many which were Baldrige quality award

Index of Authors of

"Managing Information about Processes"

- Thomas H. Davenport
- Michael C. Beers

Figure 5: Original RMM is limited in its functionality. The user would have to navigate from page a), which contains only the titles of the articles classified under "business process reengineering" to screens b) and c) to find out the names of the authors of that article. The extensions to RMM we present here enable more intuitive designs.

3.2 Lack of Context in Navigation (Limitation 2)

As a consequence of poor screen contents, navigation could result in loss of context. For example, Figure 5 illustrates the navigational path a user is forced to take to find the names of the authors of an article. The user has to navigate two hyperlinks in order to reach the screen Figure 5-c, which at that point is devoid of context and therefore confusing; the user may well have forgotten why she wanted that information. Compare this to Figure 6, which shows what can be accomplished by extending RMM as proposed in this article. Now, information appears in context, and the author pages, the article page and the issue page are all only one hyperlink away.

3.3 Top-Down Design Only (Limitation 3)

The third limitation of RMM we address in this paper relates to its inadequacy to support bottom-up design. The original RMM did not have modeling primitives that would allow one to build an application from smaller components. We could not, for example, decide what information to include in a bibliographic reference page before having designed the complete RMM diagram. Nor could we, for example, merge such bibliographic references within an index to create the screen shown in Figure 6. The difficulty lies in the lack of modeling primitives to design smaller pieces without having completed the overall design.

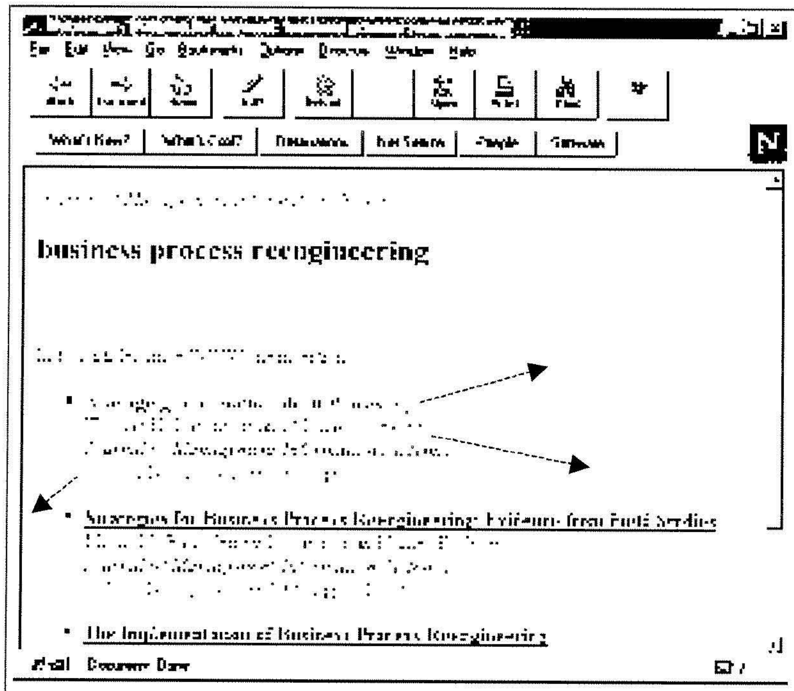

business process reengineering
Strategies for Business Process Reengineering: Evidence from Field Studies
The Implementation of Business Process Reengineering
Document: Done

Figure 6: The arrows indicate m-slice attributes that would have been separate pages in the original RMM. The use of m-slices provides for aggregation and more flexible navigation.

3.4 Loss of the Big Picture (Limitation 4)

A fourth limitation we address here relates to a shortcoming of the m-slice extensions presented in [Isakowitz et al. 1997a]. By adopting a bottom-up only approach, one loses the ability to model in a single diagram the complete WIS application. This is a significant loss because it brings us back to page-oriented web design. A solution to this, first presented in [Isakowitz et al. 1997b], involves an algorithm that recovers the global, top-down view from the bottom-up approach. We refine that contribution in this article and integrate it with the remaining RMM extensions.

4. The Extended RMM Data Model

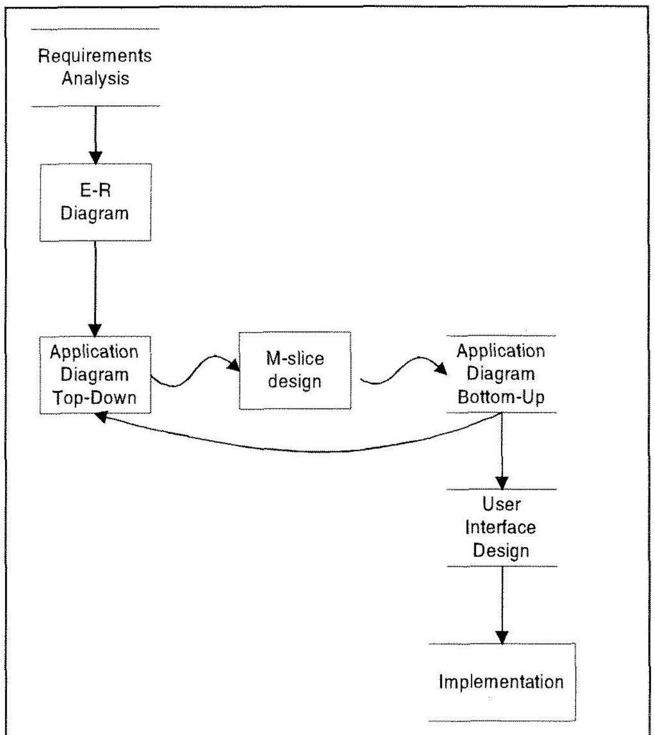


Figure 7: New design process in RMM

In response to the limitations of RMM described in the previous section, we have extended the methodology by introducing two new constructs and by updating the whole design process from requirements analysis to implementation. The first construct we introduce is the *application diagram,* which depicts the entire application (addressing limitation 4) and embraces both a top-down and a bottom-up approach (addressing limitation 3). The more powerful and flexible application diagram replaces RMM's original navigational RMM diagram. The second construct we introduce is the *m-slice,* a more powerful version of the *slice* in the original RMM. The most important characteristic of an m-slice is its ability to combine elements from any entities in the E-R diagram (addressing limitation 1), thereby providing flexible aggregation and contextualized navigation (addressing limitation 2). Hence, these two new constructs, which are fully described in the next sections, help solve the previously discussed limitations of the original RMM.

Using the new constructs, we have updated the entire design process of RMM. Figure 7 shows the new process. The designer begins with a requirements analysis to determine the following:

- what the information domain is,

- what the application will do,
- who will use the application, and
- how the users will use the application.

Next, the designer creates the E-R diagram, based on the familiar principles of E-R design, to model the information structure of the application. The following step involves designing the application diagram in a top-down fashion. The designer then decomposes the application diagram into its building blocks, the m-slices, and designs each one separately. By combining all the m-slices, the designer then generates the application diagram in a bottom-up fashion and compares the two versions, thereby debugging and refining the diagram in an iterative fashion. After the m-slices and application diagram have been reconciled, the designer designs the user interface and implements the design using any available tools. The next few sections will cover in detail the steps in the iterative cycle (application diagram top-down, m-slice design, application diagram bottom-up, resolve differences, repeat) and will

briefly cover the implementation step.

4.1 Designing the Application Diagram Top-Down

After designing the E-R diagram (see Figure 8), the designer can move to the next step in the RMM design process: the top-down *application diagram*. The application diagram provides a global view of the hypermedia application and is a representation of how the designer and the users envision the final application. The simple and flexible nature of the diagram allows it to be used when negotiating the application design with the users and other interested parties. As we will describe later, it can also be used in an iterative fashion to find bugs and omissions, and to refine the application in general.

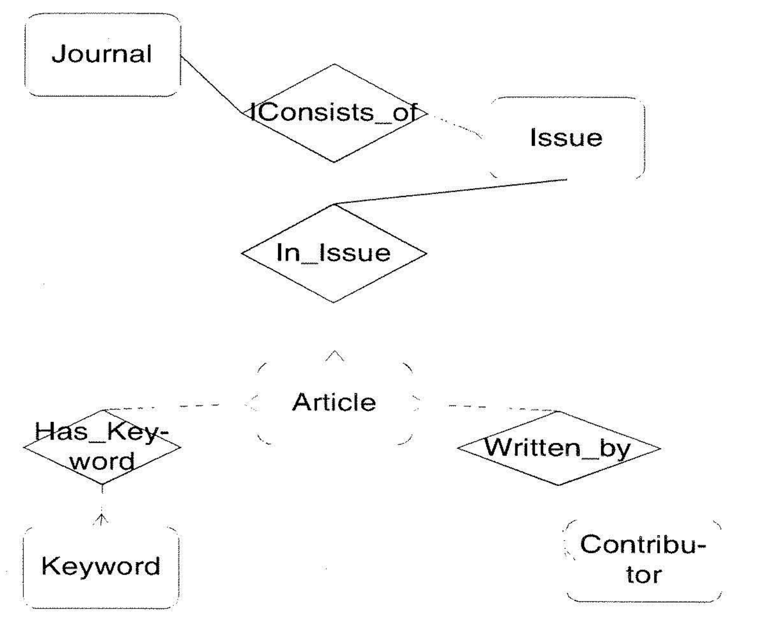


Figure 8: E-R diagram for the JMIS web site.

After consulting with all the interested user representatives, the designer constructs an initial version of the RMM application diagram. Figure 9 shows the initial application diagram for the JMIS web site. There are seven main presentation units: *journal page, issue page, keywords index, contributors index, keyword detail page, article abstract page* and *contributor page.* Consider, for example, the *article abstract page* appearing at the bottom left of Figure 9. There is one *article abstract page* per article, each containing information about that article (title and abstract), as well as hyperlinks to its authors, to the volume the article appeared in, and to the *keyword detail page*

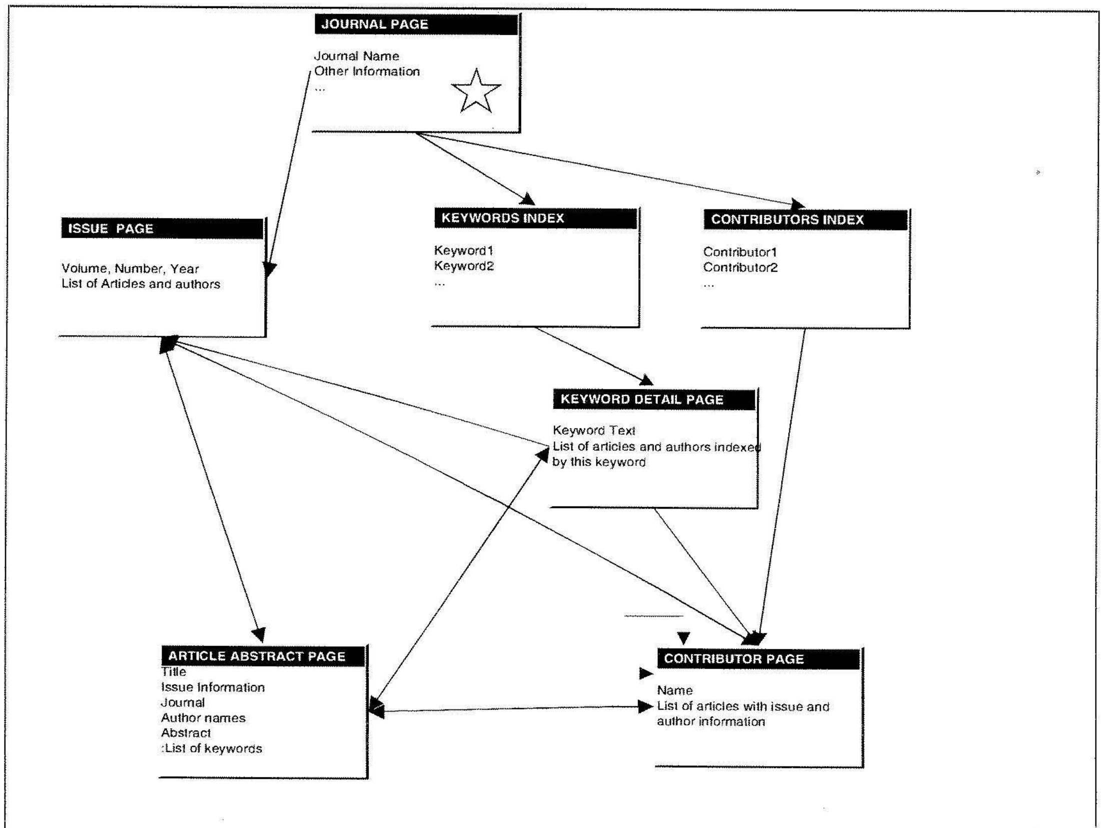


Figure 9: The initial application diagram for the JMIS Web site. All Web pages and hyperlinks among them are shown.

corresponding to the keywords that classify the article. The star that appears in the *journal page* is a notational shorthand, meaning that all other presentation units point to its presentation unit.

This initial application diagram is designed in a top-down fashion. It will be used to design the building blocks of the application, as well as for comparison with the second application diagram to be constructed in a bottom-up fashion. The main point here is that when designing the initial application diagram, the designer need not be concerned with the mechanics of each web page but only with the general structure of the application.

4.2 M-slices: the Building Blocks

When the initial application diagram is finished, the designer can start decomposing each presentation unit – a web page, for example – into smaller units, which can be reused throughout the entire application. In order to represent these units, we introduce a new construct: the *m-slice.*[3] M-slices are constructs used to group information into meaningful information units that are aggregated and nested to form higher level m-slices; that is, m-slices can contain other m-slices. M-slices are created by grouping together attributes from entities of the E-R diagram and/or previously defined m-slices. Besides fostering reuse, this approach promotes structured design. Ultimately, proceeding bottom-up, m-slices are nested in larger m-slices, culminating in presentation units, which are visually rendered. Figure 10 depicts the rendering of an m-slice for the JMIS web site that contains bibliographic information about an article and relevant hyperlinks. The instance shown here belongs to the article "Managing Information about Processes". We chose to group information into this m-slice because we realized that it would be reused multiple times throughout the application.

[3] The "m" in "m-slice" derives from *Matreyshka,* the nested Russian doll, a symbol of the nested nature of RMM.

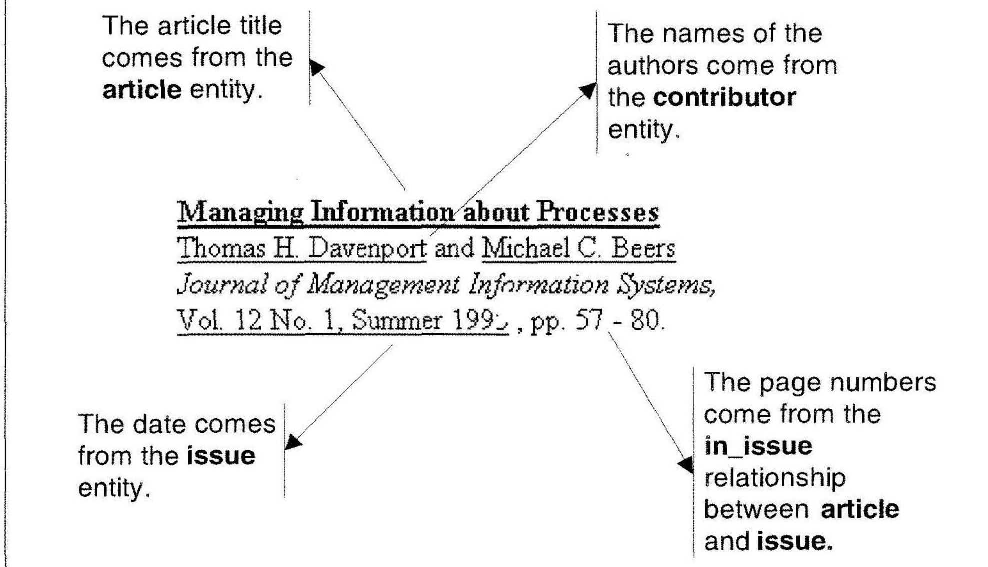

The article title comes from the **article** entity.
The names of the authors come from the **contributor** entity.
Managing Information about Processes
Thomas H. Davenport and Michael C. Beers
Journal of Management Information Systems,
Vol. 12 No. 1, Summer 1995, pp. 57 - 80.
The date comes from the **issue** entity.
The page numbers come from the **in_issue** relationship between **article** and **issue.**

Figure 10: Rendering of the m-slice for bibliographic information of articles

Each m-slice is *owned* by one specific entity from the E-R diagram.[4] The owner entity determines the instances of the attributes that are included in the m-slice. To stress the role of owner entities, m-slices are denoted by `<owner entity>.<slice name>`. In the case of Figure 10, the owner is the **article** entity, so we name the m-slice **article.bib_citation**. Note that besides elements from its owner entity (the article **title** in this case), the **article.bib_citation** m-slice also contains elements from other sources. For example, the names of the authors come from the **contributor** entity, and "Vol. 12 No. 1, Summer 1995" (the date and number of the issue in which then article was published) comes from the relevant **issue** entity instance. Thus m-slices encapsulate information from various sources: attributes of the owning entity, attributes of any related entities and access structures such as indices. Note that m-slices are not restricted to containing attributes from hierarchically related entities.

[4] As we shall see later, in some cases there is no owner, but there is never more than one.

As mentioned earlier, m-slices can be nested. For example, the **issue.date** m-slice, shown in Figure 11, aggregates the attributes **volume, number, season** and **year** of an issue and is included in the **article.bib_citation** m-slice. The **article.bib_citation** m-slice, in turn, can be used elsewhere, as in the **keyword.uses** m-slice depicted in Figure 12. This m-slice is owned by the **keyword** entity, and contains a list of all articles classified under each keyword. For each article, the information shown is its **article.bib_citation** m-slice, with all included hyperlinks.

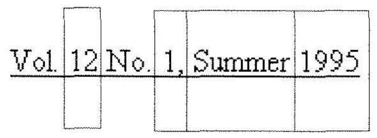
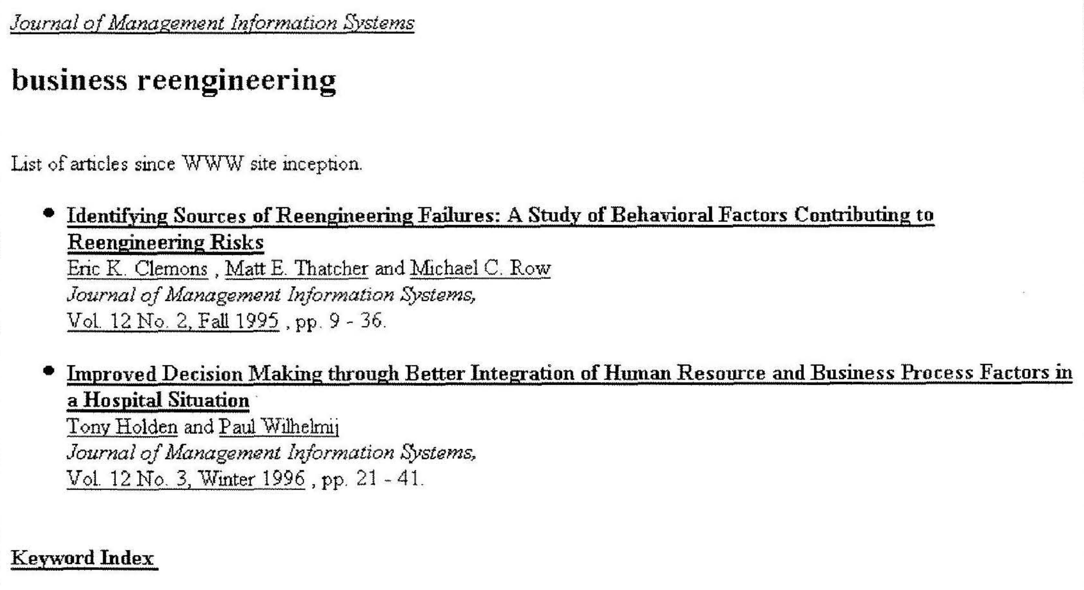


Figure 11: The issue.date m-slice



Journal of Management Information Systems

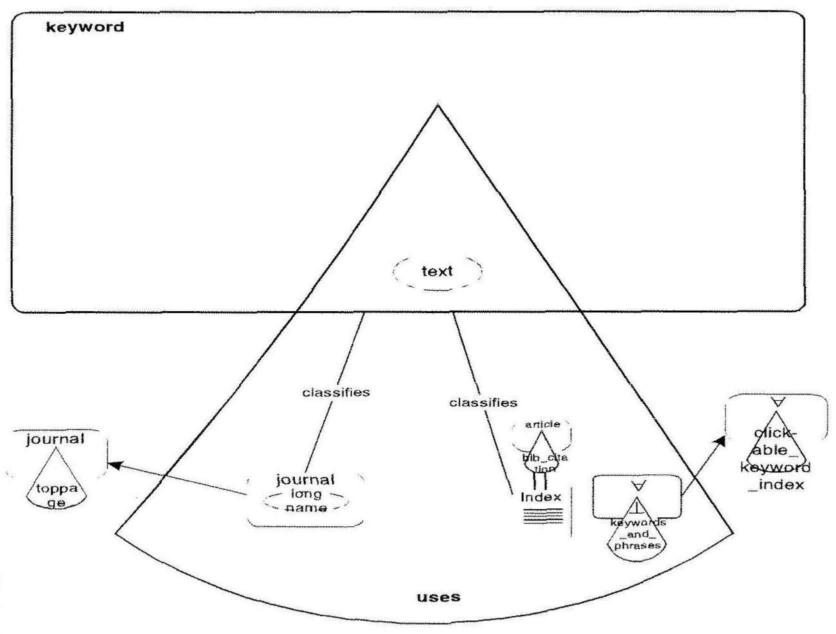
business reengineering

List of articles since WWW site inception.

- **Identifying Sources of Reengineering Failures: A Study of Behavioral Factors Contributing to Reengineering Risks**
  Eric K. Clemons , Matt E. Thatcher and Michael C. Row
  Journal of Management Information Systems,
  Vol. 12 No. 2, Fall 1995 , pp. 9 - 36.

- **Improved Decision Making through Better Integration of Human Resource and Business Process Factors in a Hospital Situation**
  Tony Holden and Paul Wilhelmij
  Journal of Management Information Systems,
  Vol. 12 No. 3, Winter 1996 , pp. 21 - 41.

Keyword Index



```
keyword.uses: m-slice
begin
      [classifies] → * journal.longname ⇒ journal.toppage;
      text;
            index begin
            relation: [classifies];
            content: article.bib_citation;
      index end;
      * ∀.keywords_and_phrases-⊥ ⇒ ∀.clickable_keyword_index
end;
```

Figure 12: The graphical and programmatic representations of the **keyword.uses** m-slice. There is 1-1 correspondence between items on the screen, in the graph and in the programming language.

We developed graphical and program specification notations to represent m-slices. (See appendices A and B for details.) Figure 12 uses both representations for the **keyword.uses** m-slice. Current research software reads the graphical notation drawn with a GUI tool and writes the program specification equivalent. This result will be input to a compiler or interpreter that associates data with the m-slices to generate the final HTML pages.

M-slices are a very important element of RMM. They allow a precise definition of information elements to be organized as a unit, while hiding (a) details - which are encapsulated in other m-slices, and (b) elements of the user-interface - for example, the format in which an index is presented, e.g., as a bulleted list or - as in the **contributor_index** case - a conjunctive list (a list of items separated by commas, with an "and" between the last two). M-slices describe *what* information is to be part of a construct and *where* to obtain it. M-slices do not describe *how* this information is to be shown. That question is addressed at the user-interface design stage. In sum, m-slices provide the power needed for RMM to represent arbitrarily complex information organizations while supporting a structured, re-usable, manageable and programmable approach to hypermedia design and development.

4.2.1 M-slice Graphical Notation

The graphical notation for the design of the structure of a hypermedia application uses many of the original primitives of RMM as well as some new ones. (For the complete list of primitives, see Appendix B.) Most of the primitives are used in the JMIS web site. In the following illustrations, we provide a rendering of the m-slice, as well as the design in graphical notation and the equivalent program specification language. A tutorial of the program specification language can be found in

The graphical notation for the m-slice, depicted in Figure 13, consists of the familiar entity and slice primitives from the original RMM, enlarged and placed so that they partially overlap. The entity portion represents the owner of the m-slice. This means that the instances of this entity determine the information that appears in this m-slice. Relationships within the m-slice use the owner entity as the source of the relationship. The name of the owner entity is found in the top left-hand corner of the entity icon.

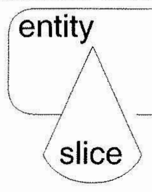


Figure 13: The m-slice primitive

4.2.1.1 Slices

The slice portion contains the constituent elements of the m-slice, whose name appears at the bottom of the slice. Note, however, that the complete name of the m-slice consists of the owner entity name followed by the slice name, e.g., **article.bib_citation**. Drawing the m-slice in this way allows us to distinguish between its physical and conceptual boundaries, as indicated in Figure 14. The slice portion defines the physical boundaries. Elements appearing within those boundaries are the ones that will be rendered in the

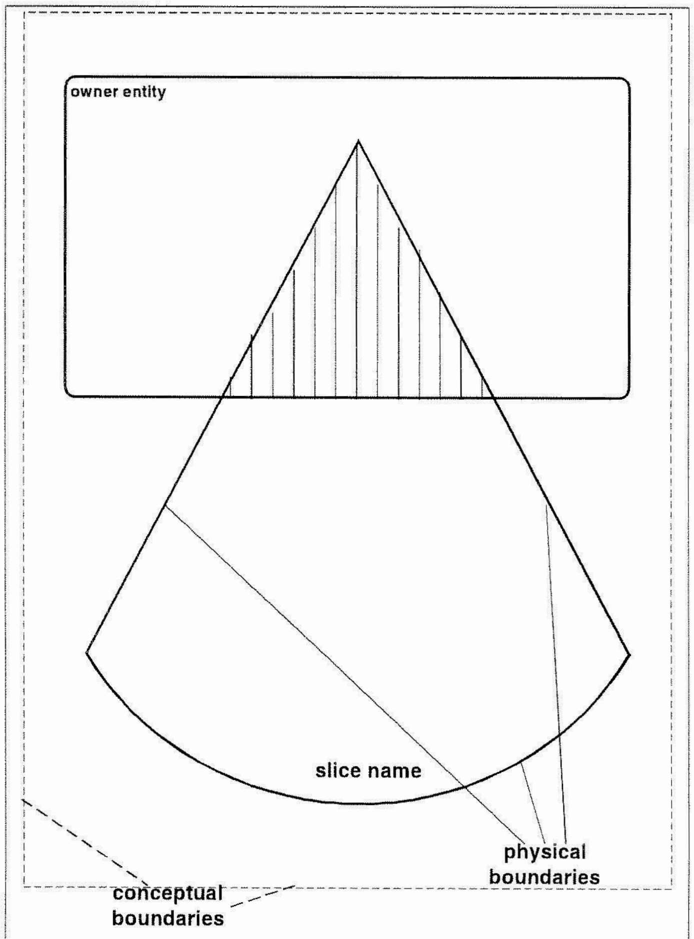
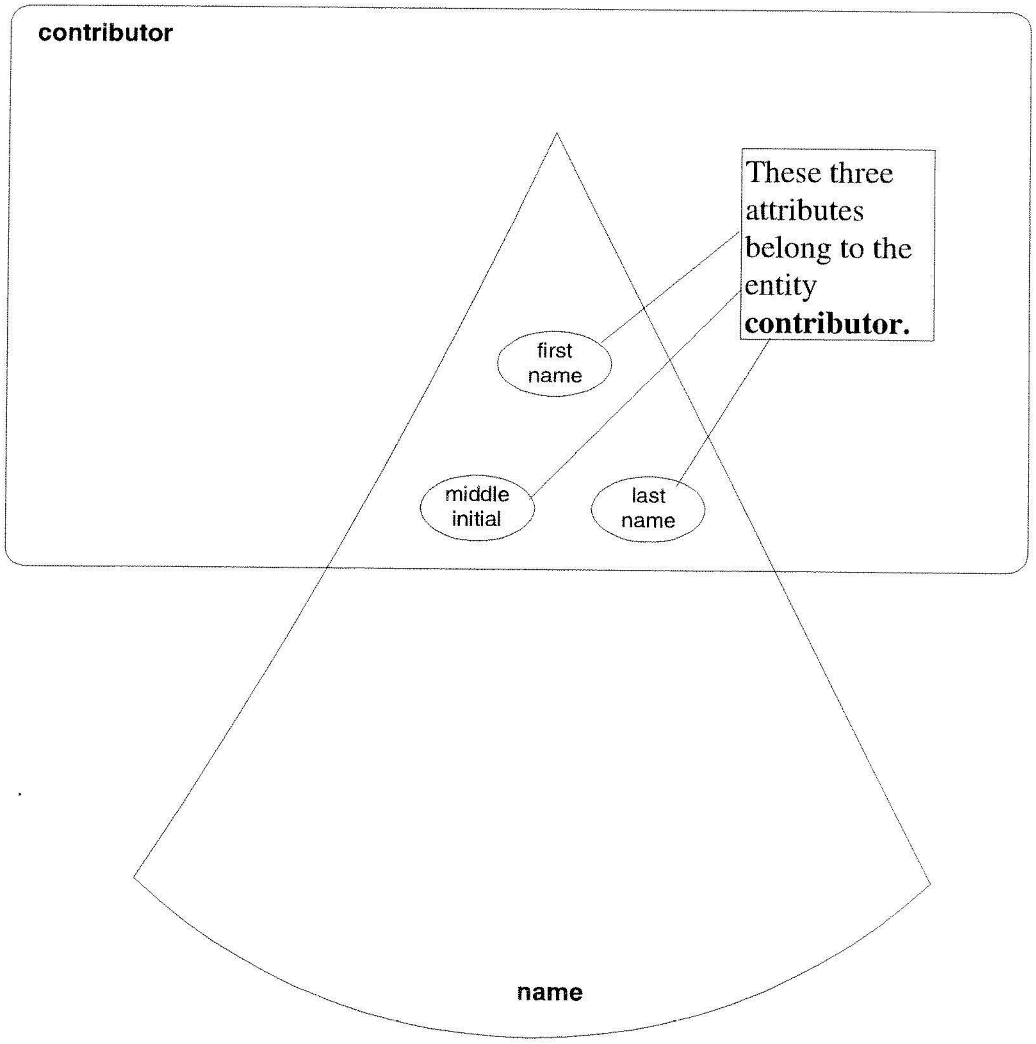


Figure 14: Physical and Conceptual Boundaries

presentation unit based on that m-slice. Everything within the physical boundaries of the m-slice is visible on a web page.

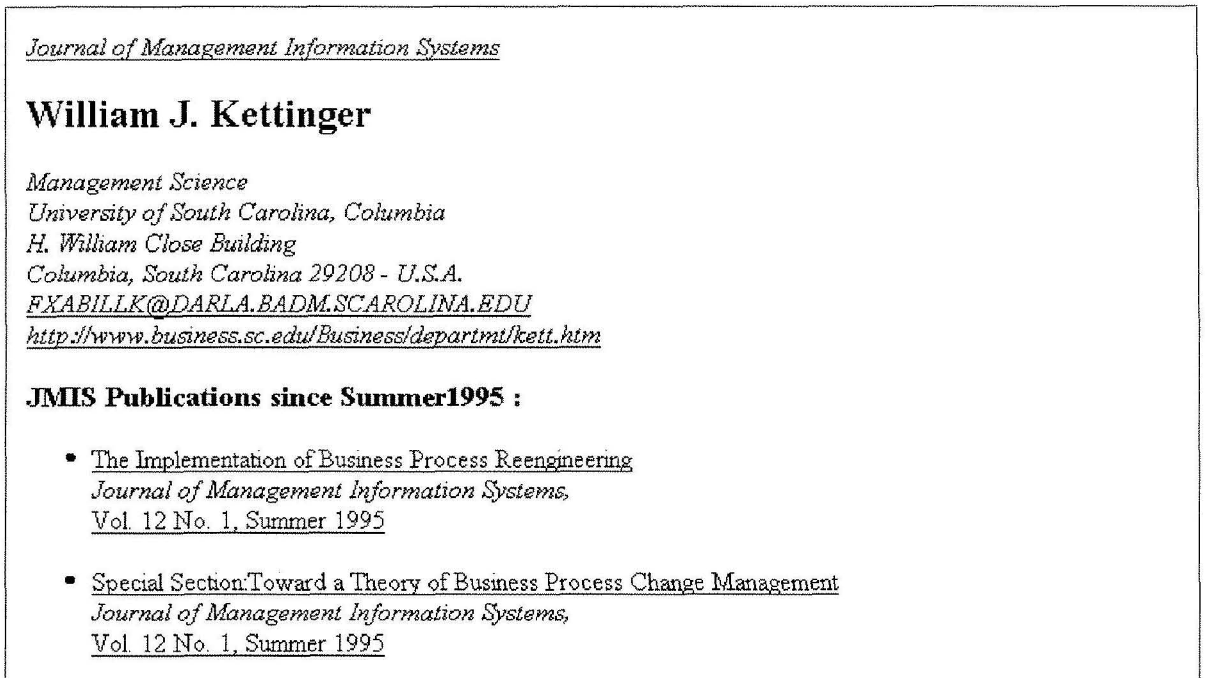
There are, however, m-slice elements that are part of the m-slice's immediate external environment but are not rendered in it, such as the destinations of hyperlinks anchored in the m-slice. We place such elements outside the m-slice's physical boundaries. These elements expand the conceptual boundaries of the m-slice (See Figure 14) beyond the physical ones. If no such elements exist, then the physical and the conceptual boundaries of an m-slice are identical. Finally, any elements (attributes or m-slices) of the m-slice that belong to the owner entity appear within the overlapping section, the shaded area in Figure 14.

Consider as an example, the **contributor.name** m-slice (Figure 15). The three attributes - **first name**, **middle initial** and **last name** - are attributes of the entity **contributor,** which is the owner entity of this m-slice. Therefore, in order to show that these three attributes are part of the m-slice **contributor.name,** we place them within the overlapping section of the m-slice. Similarly, one can use a previously defined m-slice belonging to the owner entity. For example, in **contributor.info** (Figure 16), the m-slice **contributor.name** (just described) is placed in the overlapping section to show that it is owned by entity **contributor**. Note that no entity name is used in the description of the m-slice **contributor.name** when it is placed in **contributor.info**, since the two m-slices have the same owner.


contributor
These three attributes belong to the entity **contributor.**
first name
middle initial
last name
name

```
contributor.name: m-slice
begin
      first_name;
      middle_initial;
      last_name;
end;
```

Figure 15: The contributor.name m-slice


Journal of Management Information Systems

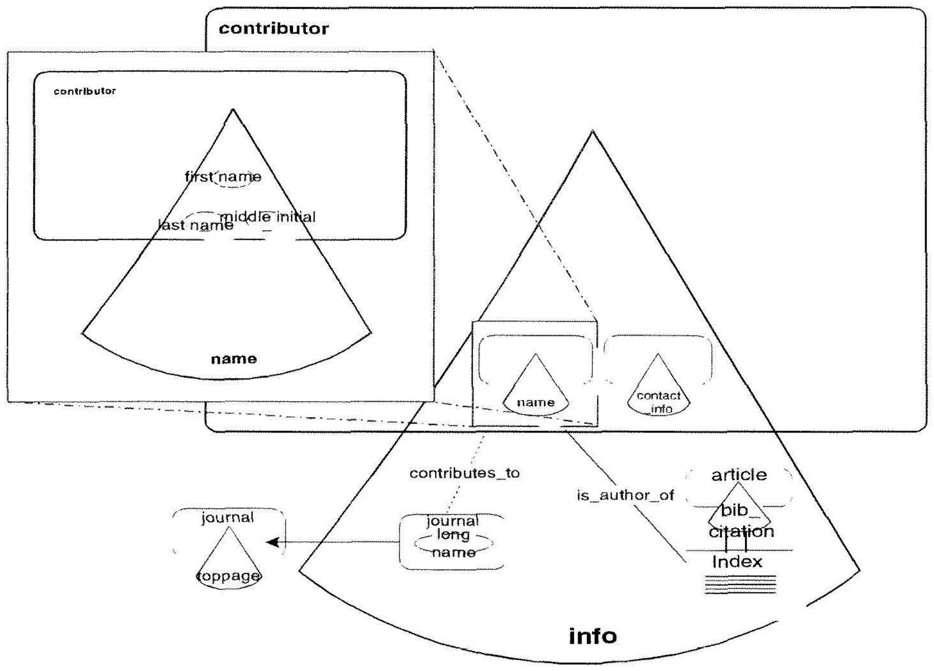
William J. Kettinger

Management Science
University of South Carolina, Columbia
H. William Close Building
Columbia, South Carolina 29208 - U.S.A.
FXABILLK@DARLA.BADM.SCAROLINA.EDU
http://www.business.sc.edu/Business/departmt/kett.htm

JMIS Publications since Summer1995 :

- The Implementation of Business Process Reengineering
  Journal of Management Information Systems,
  Vol. 12 No. 1, Summer 1995

- Special Section:Toward a Theory of Business Process Change Management
  Journal of Management Information Systems,
  Vol. 12 No. 1, Summer 1995



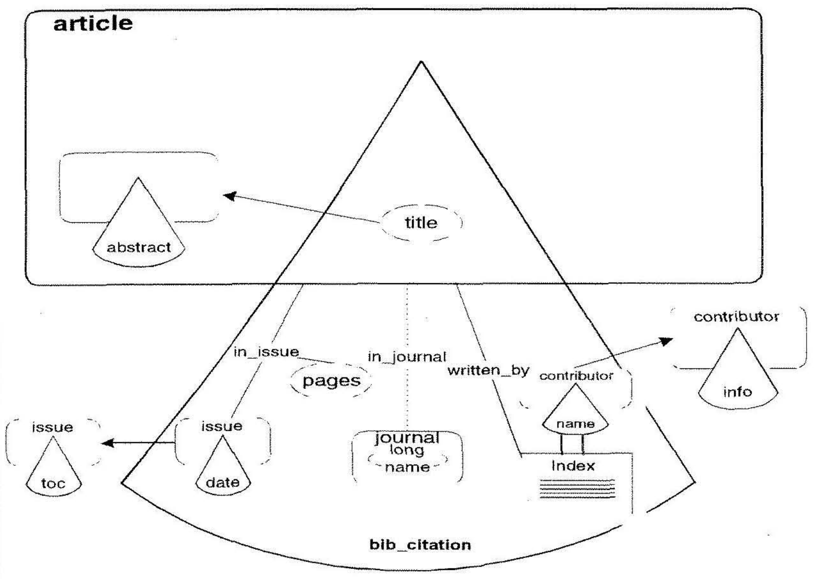
```
contributor.info: m-slice
begin
        [contributes_to]→ * journal.longname ⇒ journal.toppage;
        name;
        contact_info;
        index begin
                relation: [is_author_of];
                content: article.bib_citation;
        index end;
end;
```

Figure 16: The contributor.info m-slice contains an example of a nested m-slice (contributor.name).

Figure 16 illustrates how m-slices can be nested, making the design of the structural user interface more modular and flexible. One can "explode" any nested m-slice to show its complete structure.

An m-slice can also contain parts of entities from other than the owner entity. Those are placed in the part of the large slice that does not overlap with the owner entity. For example, notice how in Figure 17 the m-slice **issue.date** is nested in **article.bib_citation.** Since it is not part of the owner entity **article**, the notation indicates both the owner entity **issue** as well as the slice's name **date**. A shorthand notation, depicted in Figure 18, is used for an m-slice that contains a single attribute, e.g., the m-slice **journal.long_name** in

- **Managing Information about Processes**
  Thomas H. Davenport and Michael C. Beers
  Journal of Management Information Systems,
  Vol. 12 No. 1, Summer 1995 , pp. 57 - 80.



```
article.bib_citation: m-slice
begin
      * title ⇒ article.abstract;
      [in issue] → * issue.date ⇒ issue.toc;
      [in journal]→journal.longname;
      index begin
            relation: [written_by];
            content: * contributor.name ⇒ contributor.info;
      index end;
      [in issue].pages;
end;
```

Figure 17: The article.bib_citation m-slice

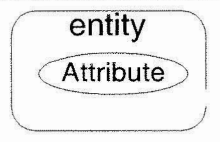


Figure 18: A single attribute m-slice

article.bib_citation, shown in Figure 17.

Whenever we use an m-slice owned by another entity we must also specify the relationship from the E-R diagram between the two owner entities. Relationships between the related entities are denoted by a line, solid for actual relationships or dotted for inferred relationships. An example of an actual relationship in **article.bib_citation** is **in_issue,** whereas an example of an inferred relationship is **in_journal.** Relation **in_issue** appears in the original E-R diagram, whereas **in_journal** is derived by combining two actual relationships: **in_issue** (between article and issue) with **in_journal** (between issue and journal).The name of the relationship, taken from the E-R diagram, is also indicated. The relationships always have the owner entity of the m-slice as their source, depicted as lines starting from the owner entity's border.

M-slices can also include access structures, such as indices. When an index is used, the relationship on which the index is based is indicated by a straight or dotted line. Additionally, it is necessary to indicate the content of the index in the m-slice. Whether it is another m-slice or a single attribute, this content determines what will be used in the m-slice, based on the relationship that the index represents. For example, in **article.bib_citation,** the index represents the relationship **written_by**. This relationship has as its source the entity **article** and as its target the entity **contributor**. The content of the index is the **contributor.name** m-slice, which is connected to the index by two lines. When this index is rendered, the entries in it are contributor names.

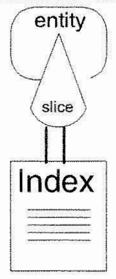


Figure 19: Index with content m-slice

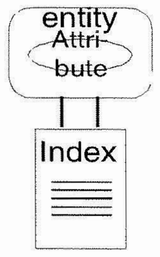
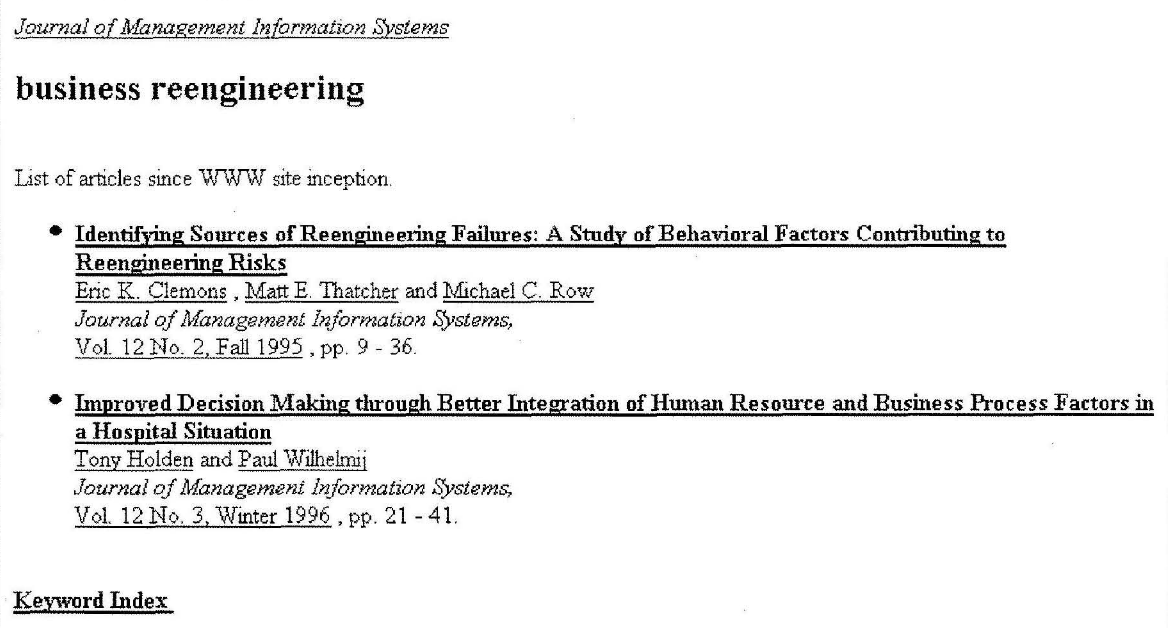


Figure 20: Index with content attribute



Journal of Management Information Systems

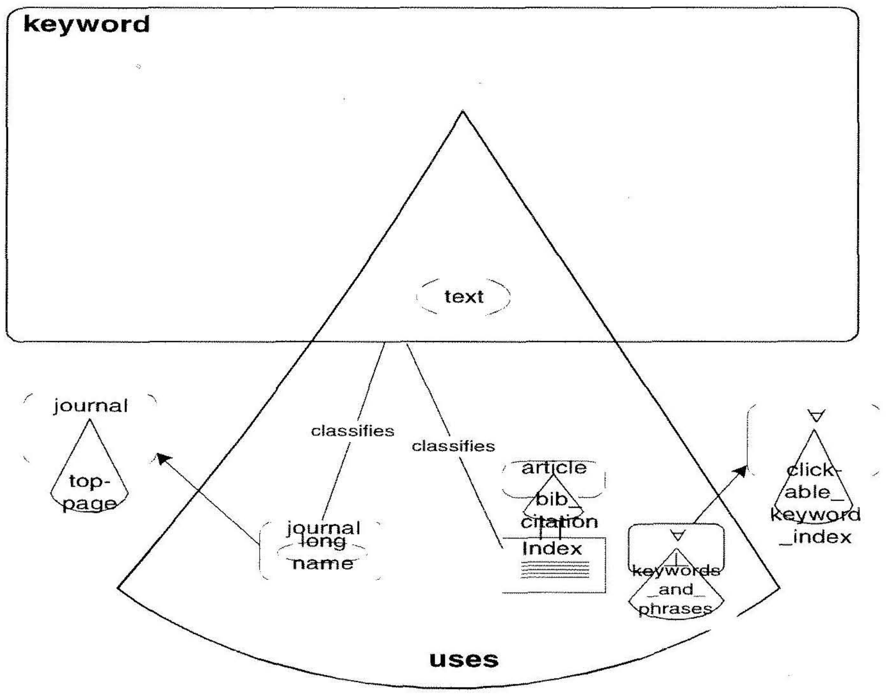
business reengineering

List of articles since WWW site inception.

- **Identifying Sources of Reengineering Failures: A Study of Behavioral Factors Contributing to Reengineering Risks**
  Eric K. Clemons , Matt E. Thatcher and Michael C. Row
  Journal of Management Information Systems,
  Vol. 12 No. 2, Fall 1995 , pp. 9 - 36.

- **Improved Decision Making through Better Integration of Human Resource and Business Process Factors in a Hospital Situation**
  Tony Holden and Paul Wilhelmij
  Journal of Management Information Systems,
  Vol. 12 No. 3, Winter 1996 , pp. 21 - 41.

Keyword Index



```
keyword.uses: m-slice
begin
      [classifies] → * journal.longname ⇒ journal.toppage;
      text;
            index begin
            relation: [classifies];
            content: article.bib_citation;
      index end;
      * ∀.keywords_and_phrases-⊥ ⇒ ∀.clickable_keyword_index
end;
```

Figure 21: The keyword.uses m-slice

In some cases, m-slices contain attributes that belong not to entities but to relationships. Thus relationships can also own m-slices. For example, in **article.bib_citation, pages** is an attribute of the relationship **in_issue** between the entities **article** and **issue**. In cases such as this, the attribute (or m-slice) is connected by a solid line to the relationship that owns it.

4.2.1.2 Hyperlinks

M-slices can also include hyperlinks. A hyperlink consists of its anchor content, the starting point, and its target, the ending point. Information in an m-slice can be used as the anchor of a hyperlink to another location within the same domain, i.e., the same web site, or outside the domain, i.e., to another web site. A hyperlink is depicted as an arrow that crosses the physical border of the m-slice. If the target is in the same domain, then it is shown as another m-slice that is defined elsewhere. In **article.bib_citation,** the m-slice **issue.date** serves as the anchor of a hyperlink which has as its target the m-slice **issue.toc**. This is shown by a uni-directional arrow crossing the border of the m-slice **article.bib_citation** and connecting the two m-slices. The same is done for the index of the relationship **written_by,** in which the content slice, **contributor.name**, now becomes an anchor hyperlinked to the m-slice **contributor.info**.

In **article.bib_citation,** we can also see an example of a hyperlink to an m-slice that is owned by the same owner entity. The attribute **title** of the entity **article** serves as a hyperlink to the m-slice **article.abstract**. Since that m-slice is owned by **article,** it appears outside the physical boundaries of the m-slice **article.bib_citation** but within the entity **article**.

4.2.1.3 Empty Slices

We often need to define a hyperlink that does not use information from the domain of the application as its anchor content, but does itself constitute a reusable component. For example, we may want to use a fixed text or image as a button that can be clicked; i.e., the anchor is information-independent. We therefore need some primitive to serve as a placeholder for the hyperlink. We call this placeholder an empty slice. In the m-slice **keyword.uses** (See Figure 21), for example, we use such an empty slice, called **keywords_and_phrases**, as a hyperlink to **clickable_keyword_index**. At implementation, this empty slice is filled with the text "Keyword Index" or the text "Key words and phrases." The anchor m-slice **keywords_and_phrases** and the destination m-slice **clickable_keyword_index** can be owned by any entity. This is denoted by the symbol ∀ in the entity portion of the m-slice. As a shorthand notation, we have a special primitive for buttons (See Figure 22). Such a button can either hyperlink to another m-slice or initiate a process, e.g., a script.

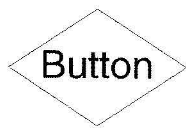


Figure 22: Button primitive

The m-slice **keyword.uses** (See Figure 21) is an example of how a complex m-slice can be used as the anchor content for an index. The index in this case is associated with the relationship **classifies** between the entities **keyword** and **article**. The entity **keyword** determines the instances of the entity **article** that are classified. We also define what information will be used for index entries: the complex anchor m-slice **article.bib_citation**. Note that the owner entity of this m-slice is **article**; if we go back to Figure 17 of **article.bib_citation**, we can see what information will be instantiated based on the current instance of the entity **article**. Since **article.bib_citation** has many hyperlinks to different m-slices, we encapsulate the internals in the definition of **keyword.uses**. If one wants to reveal the full structure of the index, one "explodes" the m-slice **article.bib_citation**.

4.3 Rebuilding the Application Diagram Bottom-Up

The next step in the RMM process is to rebuild the application diagram in a bottom-up fashion, addressing the third limitation of the original RMM. The RMM application diagram can be generated automatically from the collection of M-slices using the following two-step algorithm:

1. Examine all the m-slices used in the application.
    1.1. Note all the hyperlinks that are anchored in those m-slices. The targets of those hyperlinks are m-slices that represent presentation units in the application.
    1.2. Place all of those presentation unit m-slices in the application diagram.
2. Identify all the interactions between the presentation units placed in the application.
    2.1. Examine each presentation unit for hyperlinks.
    2.2. Place those hyperlinks in the application diagram as arrows from the anchor m-slice to the target m-slice.
    2.3. Name those hyperlinks after the relationship from the E-R diagram on which they are based.
    2.4. Examine the m-slices for any lower-level m-slices that are nested inside.
    2.5. Place in the application diagram any hyperlinks found in those nested m-slices.
    2.6. Name them after the relationship they are based on, concatenated with the relationships to the nested m-slices.

Consider, for example, the generation of the application diagram for the entire JMIS web site. The first step is to select the presentation units, i.e., the pages of the site. We examine all the m-slices that we created for the site. In every m-slice, we look for hyperlinks to other m-slices. The target m-slices of those hyperlinks are presentation units.

Looking at the m-slice **article.bib_citation** (See Figure 17), we can immediately see that there are three hyperlinks to m-slices outside this m-slice. Such hyperlinks can be identified easily since they always originate from within the large slice shape and cross its borders. The three hyperlinks in this example have as their targets the m-slices **article.abstract, issue.toc and contributor.info**. These three m-slices are placed in the application diagram as presentation units.

After examining all the m-slices of the site, we obtain the diagram shown in Figure 23. Note that we place an m-slice only once in the application diagram, even though it may be the target of multiple hyperlinks. The application diagram for the JMIS site now shows the seven pages that are used in the web site: **journal.toppage, issue.toc, contributor.info, article.abstract, keyword.uses**, **journal.contributor_index** and **journal.keyword_index**.

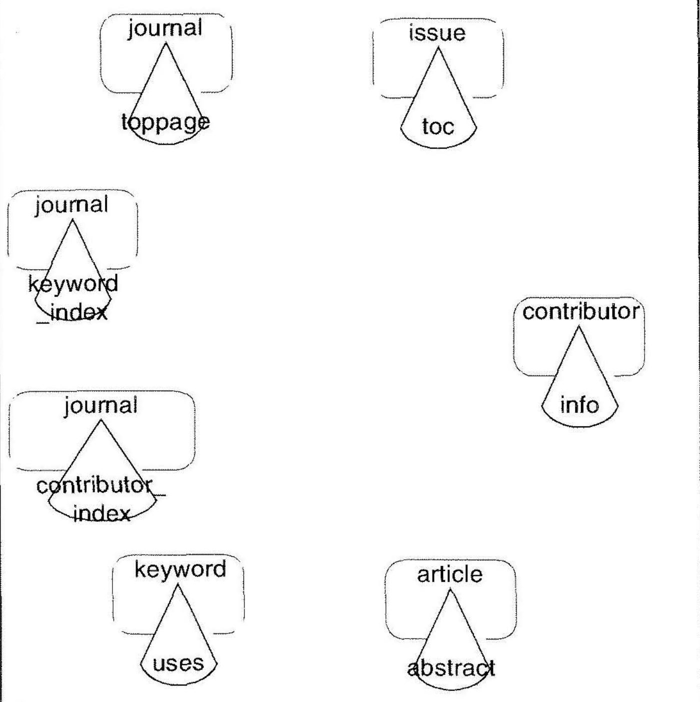


Figure 23: The first step is determining the presentation units.

The next step is to place in the application diagram the hyperlinks that exist between these pages. To do this, we consider only the m-slices that are now in the application diagram, i.e., all the m-slices that represent the web pages of the site. For each m-slice, we find all the hyperlinks to other m-slices, and we place the hyperlinks in the application diagram. Consider, for example, the m-slice **contributor.info** seen in Figure 16. We can see that there is one hyperlink to the **journal.toppage**. It is also shown from the diagram that the relationship on which the hyperlink is based is **contributes_to**, between the entities **contributor** and **journal**. We therefore name the hyperlink **contributes_to** and place it in the application diagram as an arrow starting at **contributor.info** and pointing to **journal.toppage** (See Figure 24).

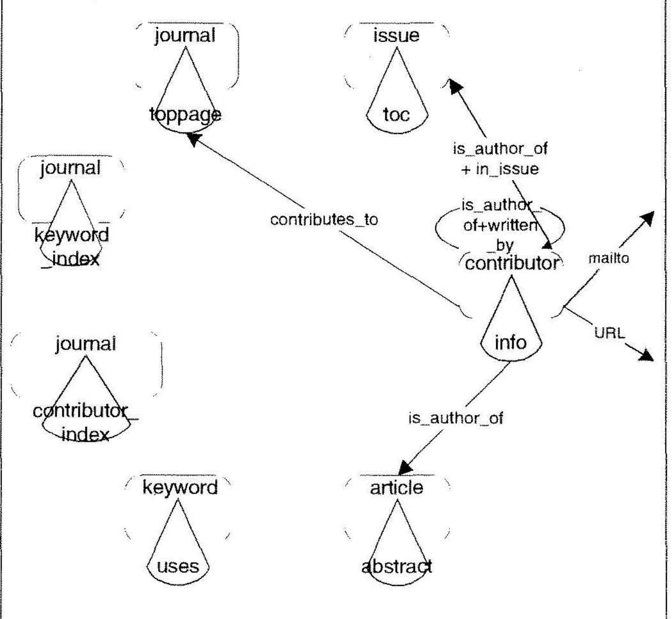


Figure 24: Adding the hyperlinks for contributor.info

After completing this process for the hyperlinks in the m-slice **contributor.info**, we recursively look for hyperlinks included in any of its nested m-slices. **Contributor.info** has four nested m-slices: **contributor.name, contributor.contact_info, journal.longname** and **article.bib_citation**. A look at **contributor.name** and **journal.longname** does not reveal any additional hyperlinks. **Contributor.contact_info**, however, has two additional hyperlinks. These hyperlinks, an e-mail hyperlink and an external hyperlink, do not point to any m-slice. We place these two hyperlinks in the application diagram, as in Figure 24. Since these are external to the site, we do not follow them.

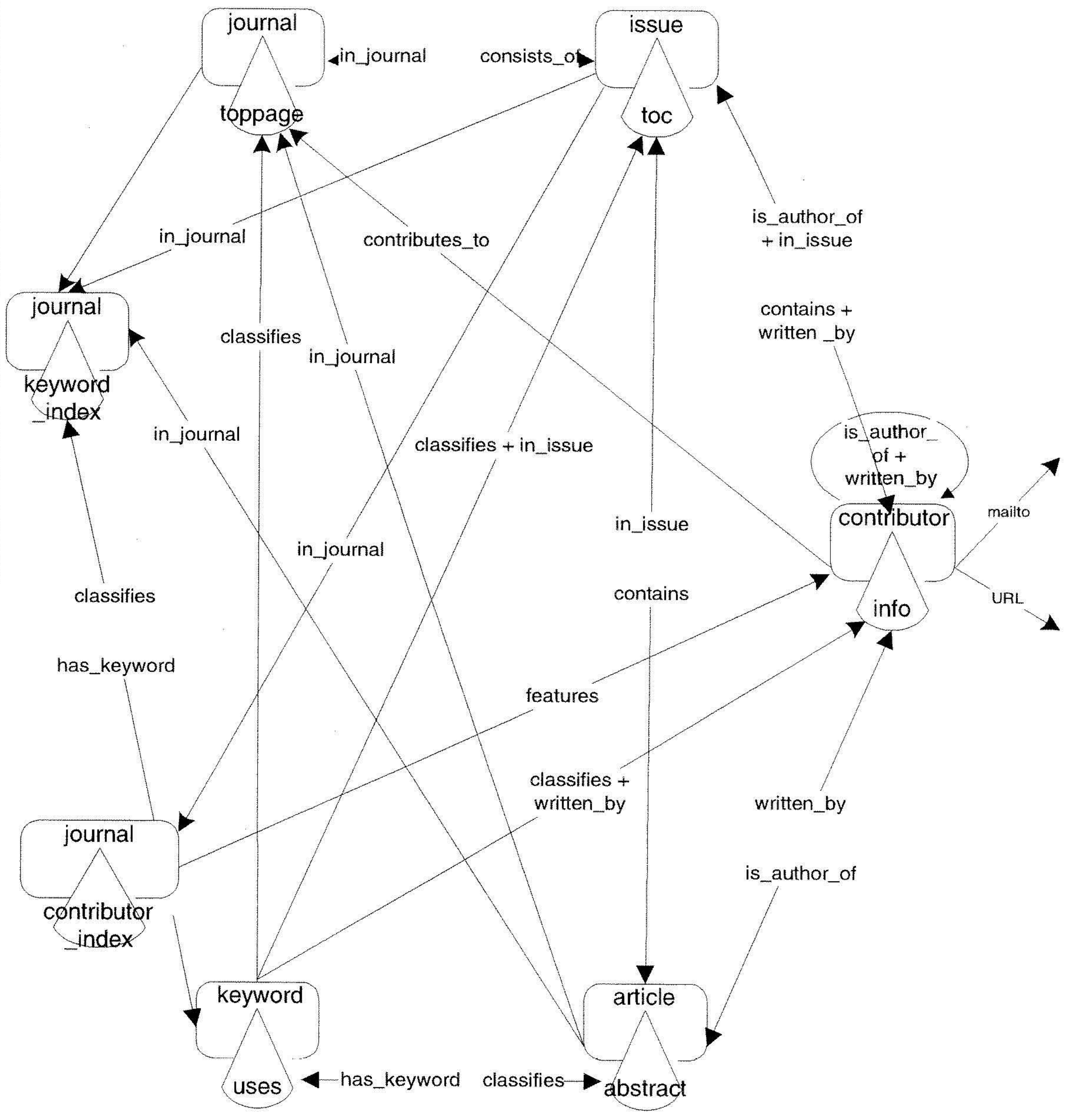


Figure 25: The complete application diagram for the JMIS web site.

Finally, a look at the **article.bib_citation** m-slice reveals three additional hyperlinks to **article.abstract, issue.toc** and **contributor.info** (a recursive hyperlink). We name these hyperlinks after the concatenation of the relationships between the hyperlink and each nested m-slice. For example, we name the hyperlink to **issue.toc** as **is_author_of + in_issue** (see Figure 24).

After going through all the pages and finding all the hyperlinks, we can construct a complete application diagram, which can be seen in Figure 25. Note that for simplicity, we use two-way arrows to denote two hyperlinks between the same m-slices. In this case, the name of each hyperlink is placed closer to the target of the arrow.

4.4 Debugging and Refining through Iteration

The similarity between the application diagrams shown in Figure 9 and Figure 25 ought to be clear. It is important to stress that while the first is created *ex ante* and thus represents a top-down approach to design, the latter is produced *ex post* from the m-slices in a bottom-up fashion. Discrepancies between the two could, for example, indicate

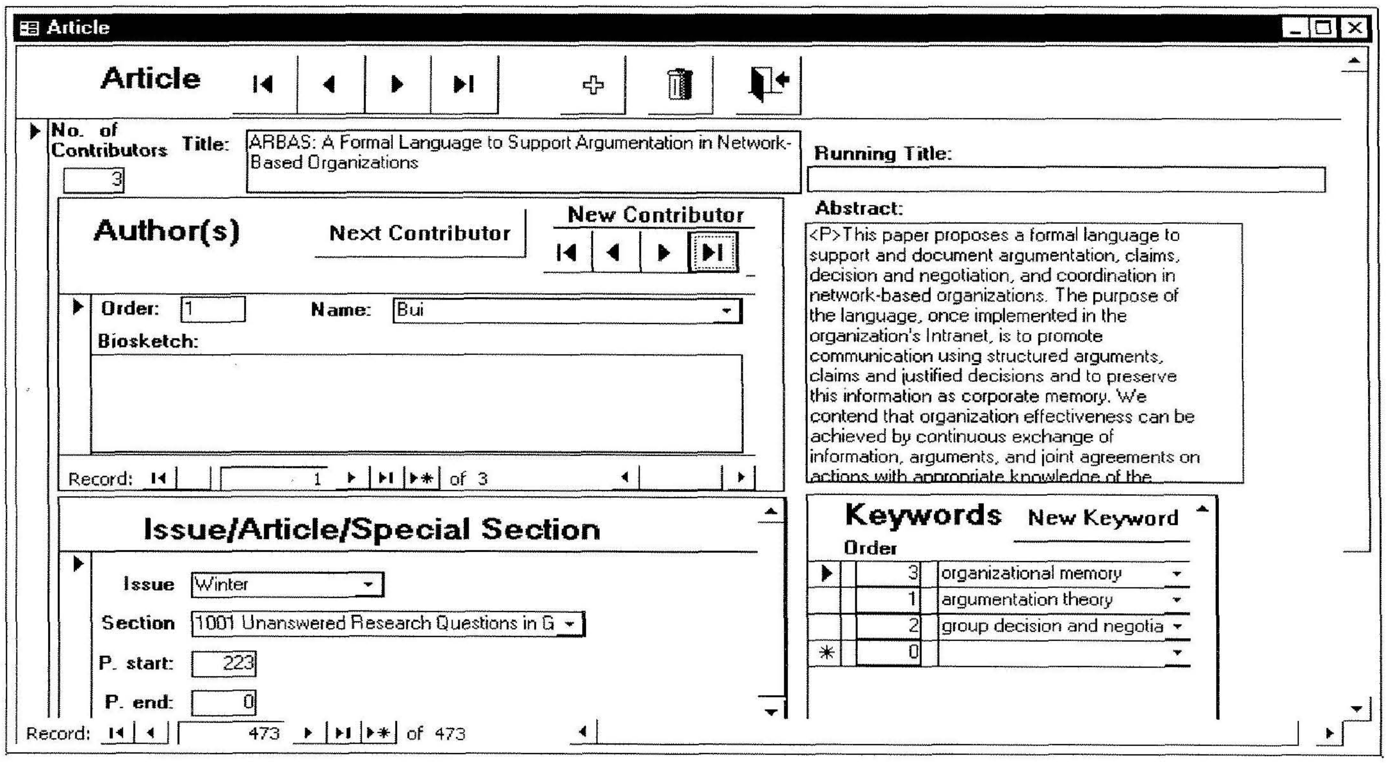
- m-slices that have not yet been implemented (those that appear in the top-down approach but not in the bottom-up),
- m-slices that ought to be either removed or hidden within higher level m-slices (those that appear in the bottom-up approach but not in the top-down), or
- missing hyperlinks.

Developers can use these two diagrams to re-shape the application by modifying the m-slices or the application diagram until the discrepancies are resolved.

This iterative nature of RMM provides flexibility to the application designer by allowing continuous refinement. Also, thanks to the intuitive nature of the application diagram, users can take part in the iterative design process. The designer can communicate easily the structure of the application to the users, who in turn can express their needs and desires. This ensures a more effective system development process.

5. Implementation

An application designed with RMM can be implemented with any database management tool or web platform. The nature of the methodology makes it independent of the implementation specifics. Also, the final application can be generated in two ways: statically or dynamically. Statically, the HTML pages that comprise the application can be generated in a simple batch process. This is more appropriate for web sites whose content changes infrequently. Since its content changes only four times a year (with each new quarterly issue), the JMIS web site is generated in this fashion, in a semi-automatic process.



No. of Contributors: 3

Title: ARBAS: A Formal Language to Support Argumentation in Network-Based Organizations

Running Title:

Abstract: <P>This paper proposes a formal language to support and document argumentation, claims, decision and negotiation, and coordination in network-based organizations. The purpose of the language, once implemented in the organization's Intranet, is to promote communication using structured arguments, claims and justified decisions and to preserve this information as corporate memory. We contend that organization effectiveness can be achieved by continuous exchange of information, arguments, and joint agreements on actions with appropriate knowledge of the

Order: 1 Name: Bui

Biosketch:

Record: 1 of 3

Issue: Winter

Section: 1001 Unanswered Research Questions in G

P. start: 223

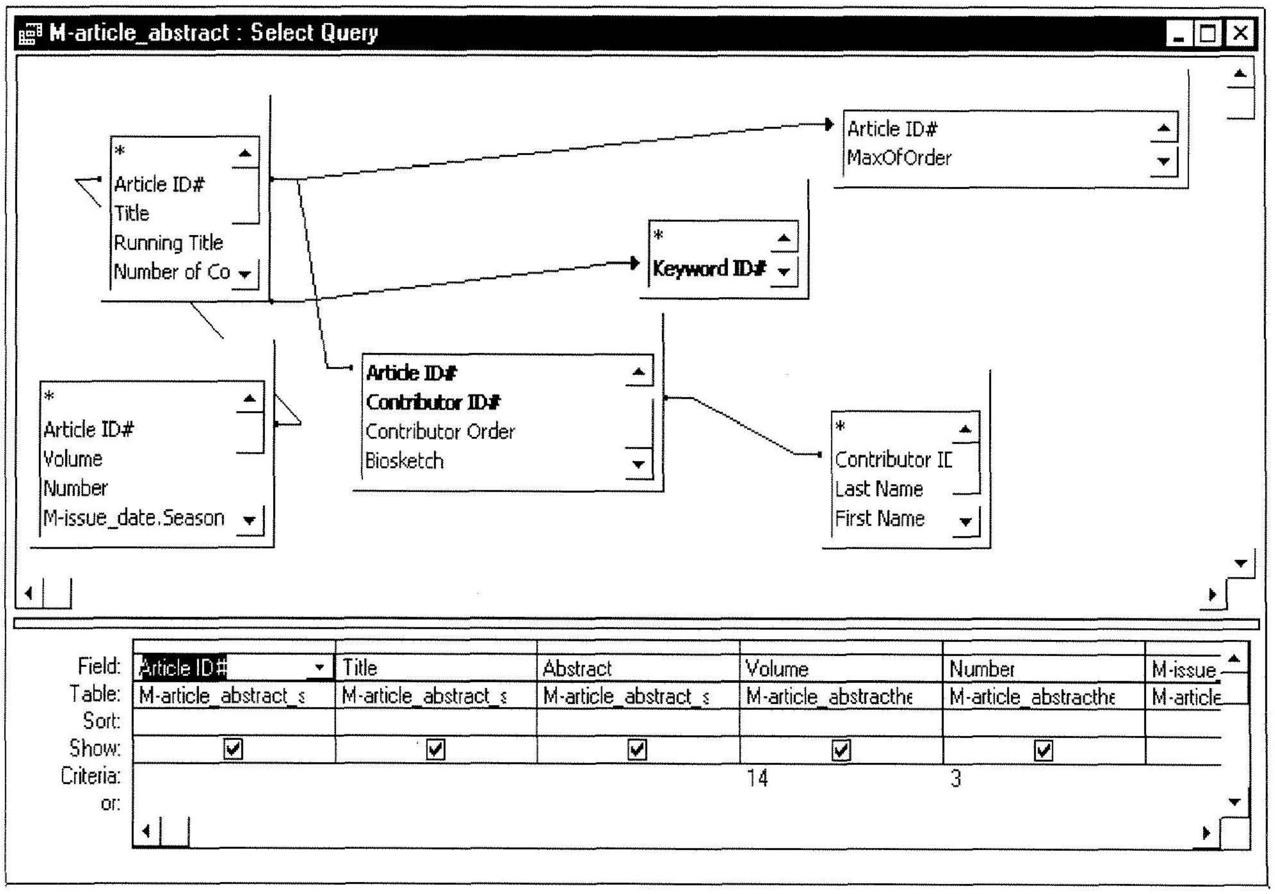
P. end: 0

	Order	
▶	3	organizational memory
	1	argumentation theory
	2	group decision and negotia
*	0	

Record: 473 of 473

Figure 26: The form used to populate the JMIS database

The JMIS site uses a relational database, implementing the E-R diagram as dictated by relational database theory. Figure 26 shows the form used to populate the database with the information on all the articles. The second part of the implementation involves creating the m-slices. Each m-slice is implemented as an SQL query. The highest level m-slices, i.e., the presentation units of the site, are

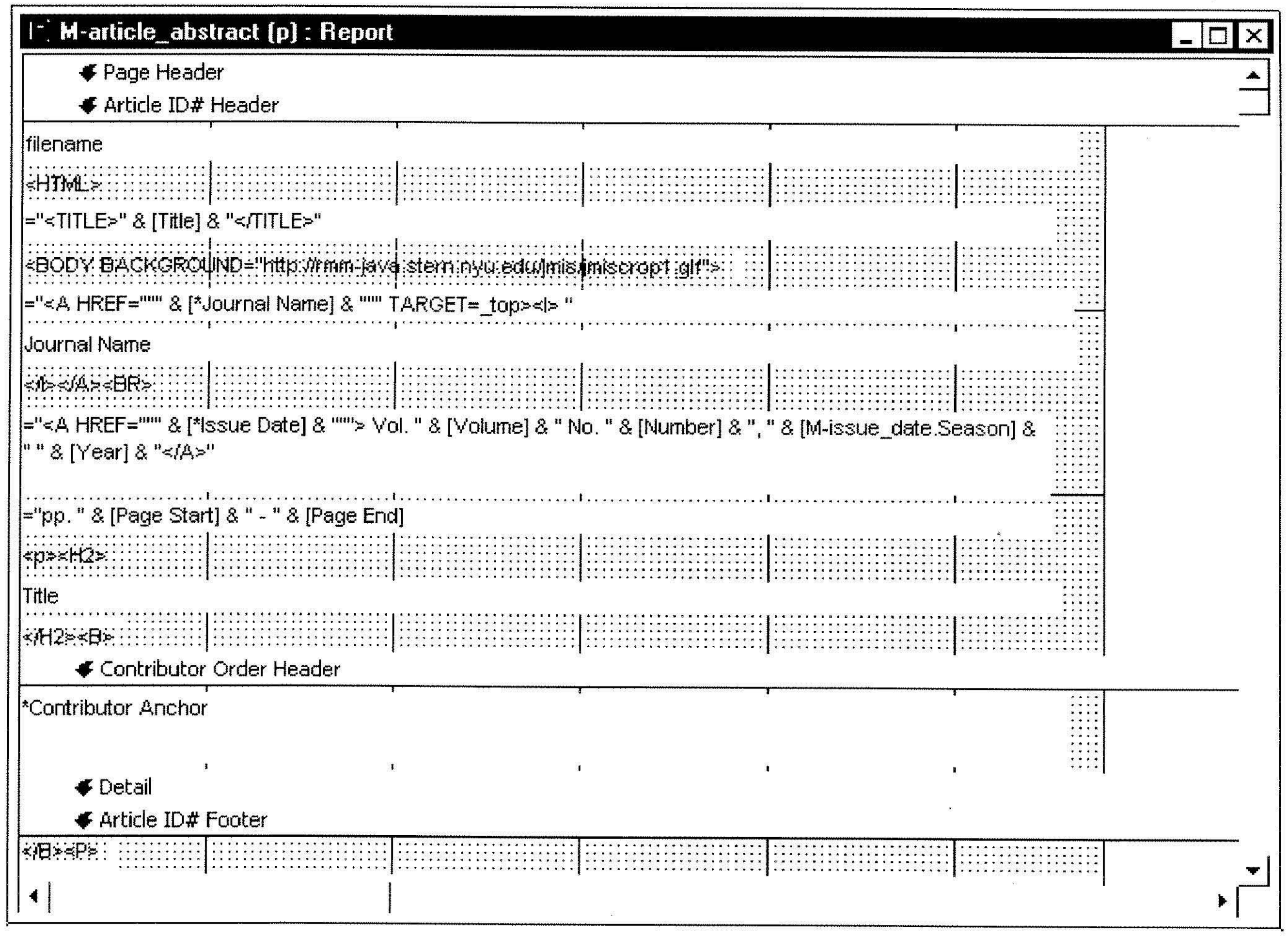


Field:	Article ID#	Title	Abstract	Volume	Number	M-issue
Table:	M-article_abstract_s	M-article_abstract_s	M-article_abstract_s	M-article_abstracthe	M-article_abstracthe	M-article
Sort:						
Show:	☑	☑	☑	☑	☑	
Criteria:				14	3	
or:						

Figure 27: The Implementation of the M-slice **article.abstract**

large queries built from smaller queries representing lower level m-slices. Figure 27 shows the implementation of the m-slice **article.abstract**, which includes the query corresponding to the **contributor.name** m-slice. These queries are then used to generate database reports. It is in these reports that we embed the HTML tags necessary for the user interface part of the application. Figure 28 shows part of the report for the m-slice **article.abstract**. Each report is converted into a text file, renamed into an HTML file, and uploaded to the server that holds the JMIS site. Updating the web site is a quick and automatic process, constrained only by the processing speed of the computer used to generate reports. Most important, any maintenance or update of the site structure is easily done via the RMM design. Additions or changes to the site are not "patches," but structured, efficient modifications.

The second way to generate the HTML pages of an application designed with RMM is dynamically. This can be done with a dynamic link between the database and the web server so that the pages dispatched by the server are designed in real-time according to the client's query specification.



```
filename
<HTML>
="<TITLE>" & [Title] & "</TITLE>"
<BODY BACKGROUND="http://rmm-java.stern.nyu.edu/mis/miscropt.gif">
="<A HREF=""" & [*Journal Name] & """ TARGET=_top><I> "
Journal Name
</I></A><BR>
="<A HREF=""" & [*Issue Date] & """> Vol. " & [Volume] & " No. " & [Number] & ", " & [M-issue_date.Season] &
" " & [Year] & "</A>"
="pp. " & [Page Start] & " - " & [Page End]
<p><H2>
Title
</H2><B>
*Contributor Anchor
</B><P>
```

Figure 28: The database report for the M-slice **article.abstract.**

This dynamic mechanism can also be used to update the content of the database, via the web site, with the use of HTML forms. This type of configuration is most appropriate for web sites whose content changes frequently and it requires a DBMS that supports dynamic querying via the Web.

6. Related Work

The research we presented here is based on the original RMM methodology introduced by Isakowitz, Stohr and P. Balasubramanian [Isakowitz et al. 1995]. For database domains, RMM has the benefit of using tools such as E-R diagrams, with which designers are already familiar. This approach is limited in many respects, particularly the ability to produce rich screens, to model navigation from a user-perspective and to support bottom-up design and development. More complex kinds of RMM screen design were presented by V. Balasubramanian, Bieber and Isakowitz in [Balasubramanian et al. 1997], who describe the concepts of *minimal* and *hybrid* slices. Minimal slices act as default anchors for hyperlinks and hybrid slices aggregate elements from different RMM elements. These concepts were further extended in [Isakowitz et al. 1997a], which describes M-slices, an extension of hybrid slices that can be nested, and in [Isakowitz et al. 1997b], which introduces the application diagram and the bottom-up RMM development process. None of those articles describes the complete, revised RMM methodology, as we do here.

Garzotto, Paolini and Schwabe's HDM data model [Garzotto et al. 1993] and its successor HDM2 [Garzotto et al. 1996] describe the structure of a database application domain adequate to support hypermedia access, but provide little support for building user views. In other words, while they describe an application domain, they do not facilitate the design and development of applications. RMM builds on HDM and HDM2 to provide the first full methodology.

Lange's EORM [Lange 1996] proposes hypermedia design methodologies based on the object-oriented paradigm. OOHDM [Schwabe and Rossi 1995; Schwabe et al. 1996] is an object-oriented hypermedia design methodology based on HDM, which incorporates some of the same functionality as RMM within an object-oriented framework. The OOHDM concept of navigational class schema, presented in detail by Schwabe, Rossi and Barbosa [Schwabe et al. 1996], is similar to RMM's m-slices. Although OOHDM has a programming-like language to describe navigational class schemas, it lacks a graphical notation. A key difference is that while RMM's m-slices define owner entities as the source of the information needed to populate the application, OOHDM's navigational class schemas lack a notion of ownership. Hence, they can be neither easily nested nor re-used via relationships. Furthermore, OOHDM lacks an equivalent to the RMM application diagram.

The W3DT (World Wide Web Design Technique) in [Bichler and Nusser 1996], which was later extended by Scharl [Scharl 1998] into eW3DT, comprises a data model and a methodology that address implementation aspects of WIS. More geared towards unstructured domains, 3W3DT uses hierarchical diagrams to represent individual pages or classes of pages within a WIS. It can also distinguish between static and dynamic pages, and has the ability to represent database interactions. Moreover, it enables reuse by allowing structures to be shared among pages. eW3DT is complementary to RMM and can be used to represent the physical structure of a web site. Thus, one can apply RMM to the high-level design, and eW3DT to model and guide the implementation.

Vigna et al. present another implementation methodology for Web applications by introducing a low-level data model implementation called WBM and a higher-level data model called WOOM which enables resource reuse and reduces maintenance costs [Vigna et al., forthcoming]. An algorithm to translate a WOOM model into a WBM is informally presented. This approach differs from e3WDT in that it follows an object-oriented approach and presents a two-tiered model. The first tier, WBM, models the physical structure of a web site, i.e., files and directories. The second tier, WOOM, is used to represent, in a more abstract way, web page constructs that are used like data structures. The model handles references (links) and containment relations among these abstract pages. WOOM is useful for modeling any kind of web site, and is proposed as an object-oriented programming language for web site development. As with eW3DT, WBM/WOOM complements RMM in that it can be used to implement an RMM design.

7. Conclusion and Future Work

In this article, we have provided a comprehensive extension to the original RMM methodology meant to address its more significant limitations. We have extended both the data model, by adding the application diagram and the m-slice constructs, and the process itself, by enabling an iterative combination of top-down and bottom-up sub-processes. We have also presented a graphical language to represent the new constructs. An obvious question at this point is *Does this represent the final RMM?* The answer is "probably no," as we continue to receive feedback and make improvements. In particular, RMM as presented here does not handle user-interaction, secondary processes (e.g., search), multiple users, or security. All of these enhancements will be presented in future publications. In addition, the automatic generation of SQL queries from graphically designed m-slices will be studied and reported in the near future. Such transformations are key for computer-aided support of the RMM design process, and will lead to the last area of innovation: the development of an improved RM-CASE tool that incorporates the features described above.

Appendix A: RMM Program Specification Language

The merits of a program specification language are that (a) it provides precise definition and (b) it provides a compilable, executable form. Here, we illustrate the programming notation with a cumulative example. Let us build the **article.bib_citation** m-slice, starting with the simplest elements.

As with any m-slice definition, we have the initial m-slice structure, as follows:

```
article.bib_citation: m-slice
begin
<body>;
end;
```

First, let us take a look at the simplest construct, the long name of the journal, "*Journal of Management Information Systems*". This is an attribute of an entity, **journal**, related by a relationship, **in_journal**. The syntax for this is as follows:

```
[relationship] → <entity>.<slice>
```

It is important here to note that in its simplest form, an m-slice contains a single attribute. In those cases, we use the attribute's name as a shorthand notation for the m-slice. In this way we avoid having to define all single-attribute m-slices. Here, the code is as follows:

```
[in_journal] → journal.longname;
```

where **longname** is the only attribute used by the m-slice.

Next, we consider the page numbers of the article within the issue. **Pages** is an attribute of neither the article nor the issue, but of the relationship between them. Syntactically, this is as follows:

```
[relationship].<m-slice>
```

and, in this example, this is

```
[in_issue].pages
```

Next, we describe the hyperlink construct. Consider the anchor “Vol. 12 No. 1, Summer 1995”, which is the **issue.date** m-slice. This anchor leads to the table of contents of the issue containing the cited article. The corresponding code is the following:

```
[in_issue] → * issue.date ⇒ issue.toc;
```

More generally, the syntax for hyperlinks is as follows:

`[`*`relationship`*`] → * `*`<anchor>`*` ⇒ `*`<destination>`*

where *`<anchor>`* and *`<destination>`* are m-slices.

No matter how complex *`<anchor>`* and *`<destination>`* are, the * and ⇒ characters tell us immediately that we have a hyperlink.

The anchor “Managing Information about Processes” illustrates another example of a hyperlink. The title is an attribute of the article, and here it serves as an anchor to that article’s abstract page. The code is as follows:

```
[this] → * article.title ⇒ article.abstract;
```

where “`this`” is a special relationship, meaning “this same entity instance”.

When an attribute (or m-slice) is of the owner entity, we use “`this`” as the relationship. To simplify the notation, we employ a shorthand, omitting the “*`[this]→<entity>.`*” prefix. Using this shorthand, the code for the title anchor becomes:

```
* title ⇒ abstract;
```

The most complex construct in the **article.bib_citation** m-slice is the list of authors who wrote the article, which introduces the index construct. An index consists of two parts, the content to be displayed (an m-slice) and a specification of the entity instances from which to draw the content. The syntax for an index is the following:

```
index begin
    relationship: <relationship name>
    content: <m-slice> | <hyperlink>
index end
```

In this example, it is

```
index begin
    relationship: [written_by];
    content: * contributor.name ⇒ contributor.info;
index end;
```

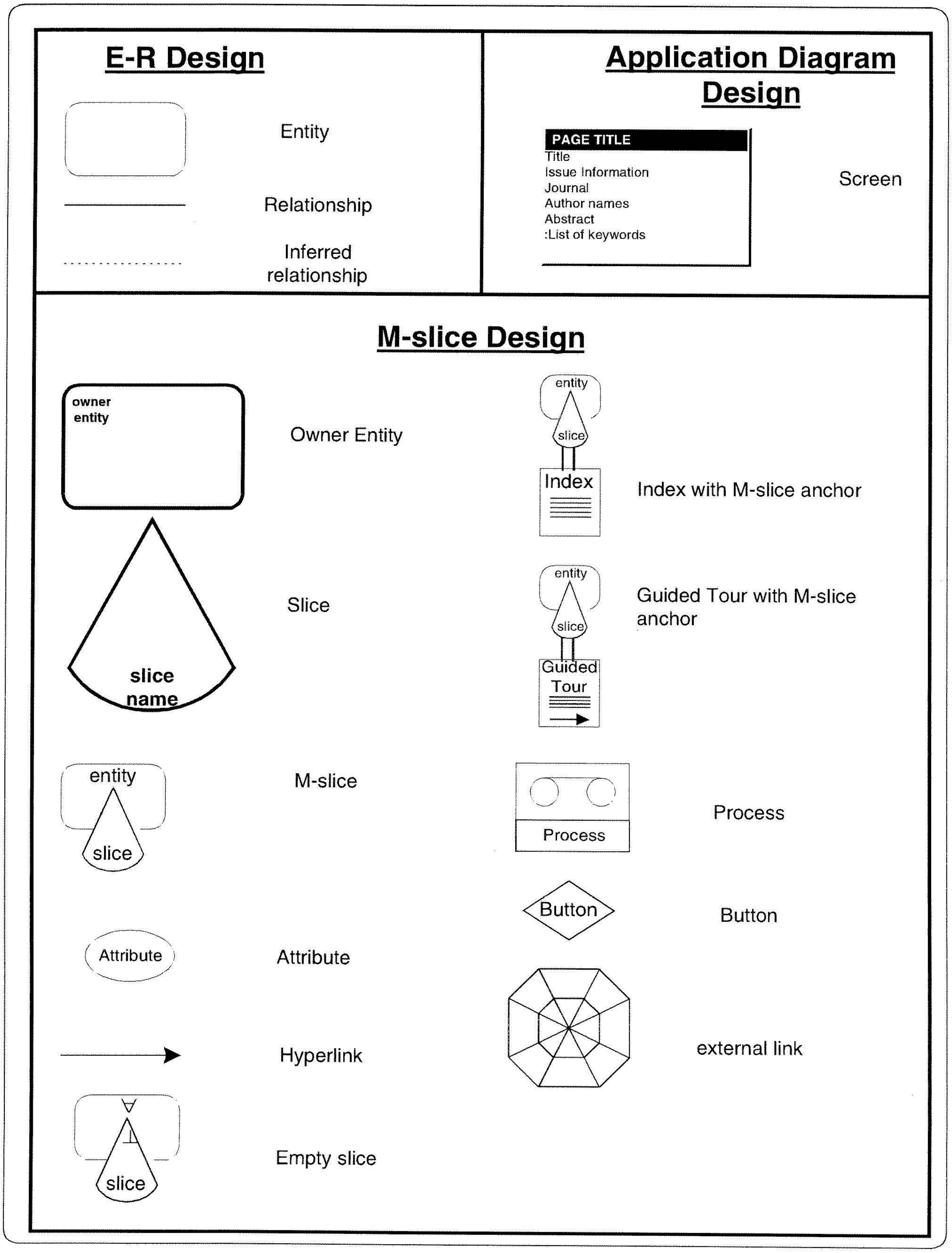
Note that the content of this index is a hyperlink from **contributor.name** to **contributor.info.** If the content were the **contributor.name** m-slice without a hyperlink, the code would be as follows:

```
index begin
    relationship: [written_by];
    content: contributor.name;
index end;
```

Putting all the pieces together, we have the following:

```
article.bib_citation: m-slice
begin
    [in_journal] → journal.longname;
    [in_issue].pages;
    [in_issue] → * issue.date ⇒ issue.toc;

    *title ⇒ abstract;

    index begin

          relationship: [written_by];

          content: * contributor.name ⇒ contributor.info;

    index end;

end;
```

Appendix B: M-slice Primitives



References

Balasubramanian, V., Bieber, M.P. and Isakowitz T. Systematic Hypermedia Design. *International Journal of Human-Computer Studies.* (1997) Also published in *Information Systems Journal,* forthcoming 1998.

Bichler, M. and Nusser, S. Modular Design of Complex Web-applications with W3DT, *Proceedings of the 5th Workshop of Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '96).* Stanford, IEEE Computer Press (1996), 328-333.

Garzotto, F., Mainetti, L. and Paolini, P. Navigation in Hypermedia Applications: Modeling and Semantics. *Journal of Organizational Computing and Electronic Commerce.* 6, 3 (1996), 211-238.

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

Isakowitz, T., Kamis, A., Koufaris, M. Extending the Capabilities of RMM: Russian Dolls and Hypertext, *Proceedings of the 30th Annual Hawaii International Conference on System Sciences,* (1997a). (Also published as Extending RMM. *CRIS Working Paper Series* # IS-96-8. Stern School of Business, NYU.)

Isakowitz, T., Kamis, A., Koufaris, M. Reconciling Top-Down and Bottom-Up Design Approaches in RMM, *Proceedings of the Workshop on Information Technologies and Systems* (WITS-97), Atlanta, GA, December (1997b), 190-199.

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

Isakowitz, T., and Bieber, M., Introdcution to Special Issue on Web-Based Information Systems, *Communications of the ACM*, July 1998.

Lange, D. B. An Object-Oriented Design Approach for Developing Hypermedia Information Systems. *Journal of Organizational Computing and Electronic Commerce*. 6, 3 (1996), 269-294.

Nielsen, Jakob. Sun's New Web Design, *http://www.sun.com/980113/sunonnet/*, January 13, 1998.

Scharl, Arno. Reference Modeling of Web Information Systems using the Extended World Wide Web Design Technique, *Proceedings of the 31st Annual Hawaii International Conference on System Sciences (HICSS-31)*, Big Island of Hawaii, Hawaii. (Jan. 1998).

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

D. Schwabe, G. Rossi and S. Barbosa. Systematic Hypermedia Design with OOHDM. *Proceedings of the ACM International Conference on Hypertext (Hypertext '96)*, Washington, DC. (March 1996).

Vigna, Coda, Garzotto, F., and Ghezzi. A Generative World Wide Web Object-Oriented Model, forthcoming.