Extending the capabilities of RMM:

Russian Dolls and Hypert

Tomás Isakowitz, Arnold Kamis and Marios Koufaris
Stern School of Business
New York University
44 W. 4th St.
New York, NY 10012

tomas@stern.nyu.edu

Extending the capabilities of RMM:

Russian Dolls and Hypertext

ABSTRACT

Hypermedia design is usually *ad hoc*. Whereas the original Relationship Management Methodology (RMM) provides a structured approach to design and implementation of hypermedia applications, it has limitations that constrain the usability of the kinds of applications it can construct. This paper provides extensions to RMM that enable it to model a much richer class of applications. Thereby making the methodology more attractive for software developers to use. The paper also presents a graphical and programming language notation for RMM's new *m-slice* construct, which is at the core of the extensions presented here.

1. Introduction

The problem with much hypermedia design is that it is *ad hoc*. RMM (Isakowitz et al., 1995) provides an effective, structured design methodology for the development of applications that are easier to maintain and extend (see Appendix C for a brief description of RMM). Currently, RMM is in use at Merryl Lynch, at publishing houses (M.E. Sharpe, Inc.), research institutions (personal contacts at Bellcore), and by educational institutions to deliver on-line teaching materials (Pace University in NY; SYRECOS consortium in Luxembourg, Staffordshire University in the UK).

After some initial experimentation with RMM, it became clear to us that its data model is limited. RMM does not account for issues beyond the basic navigational structure of a hypermedia application. Most important, it does not allow for rich information to be effective, structursity RMM, one can design pages using pure RMM concepts, but non-trivial pages quickly become awkward. This happens, for example, when information from different entities needs to be ityglayed within a single screen. Although RMM can currently model the basic navigational structure of the web site it cannot combine various components in meaningful ways.

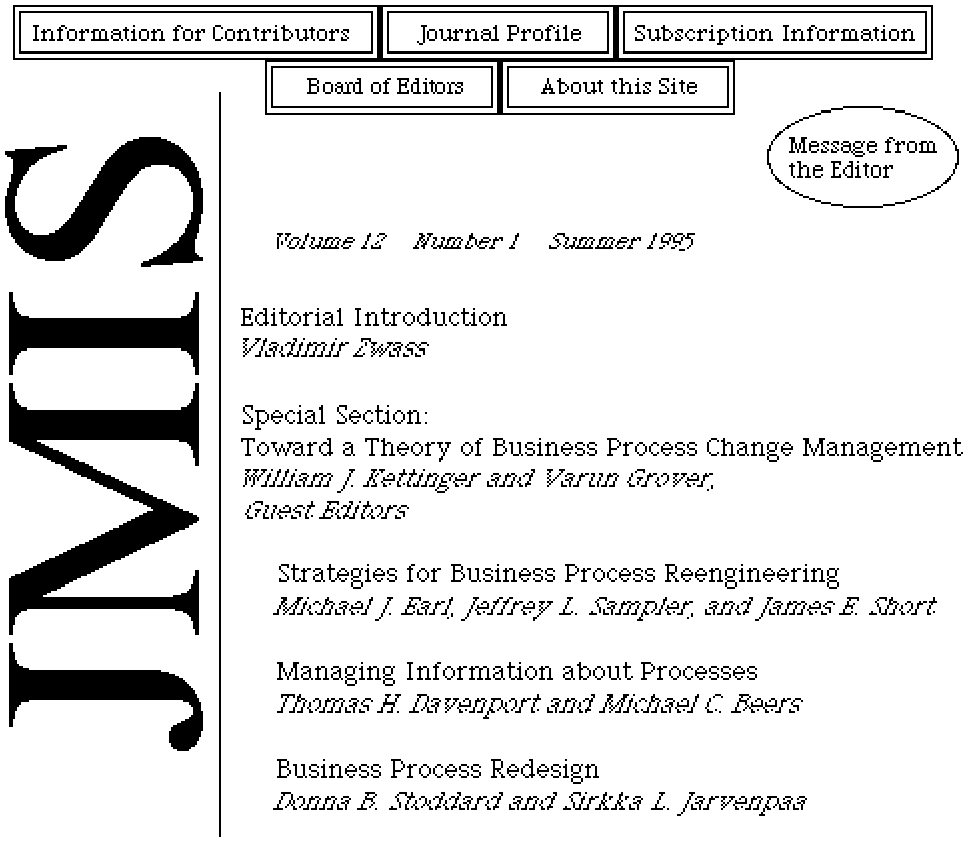
This paper extends RMM by introducing new constructs that overcome these limitations and i˝-3 -2ructaal a web nucihing muı˝(itiokds ten dNf a hypermedia applicatways.)Tjı˝T*ı˝-0.003 T extensions we provide here support the structured design of hypermedia applications of arbitrary levels of complexity.

In the paper, we first describe the web site for the Journal of Management Information Systems (JMIS) that we developed using the current RMM methodology. We describe the limitations we discovered in RMM and then propose the m-slice construct as an extension to the RMM data model to overcome those deficiencies. We then present a graphical notation and a program specification language for the new constructs. We conclude the paper with a discussion of related work and a summary of our findings.

2. RMM Application: The JMIS Website

The Website for the Journal of Management Information Systems (http://www.stern.nyu.edu/jmis) was designed to mirror the appearance of the physical journal. As shown in Figure 1, each issue has its own page with an index of all the articles and authors (table of contents) featured in that issue as well as a number of buttons 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 Website.

There is a separate page for each article in which the abstract, its keywords, and the authors' names are provided. The author names are linked to pages containing rich information about each author. There is also a link 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 linked to the appropriate pages.



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 1: The issue table of contents page.

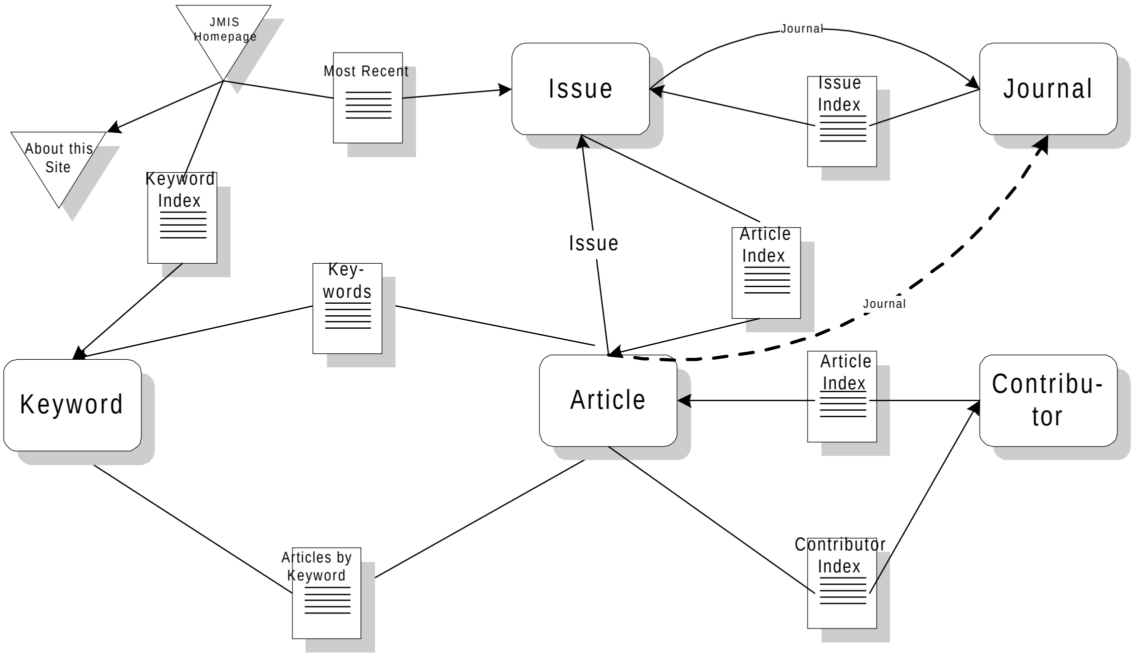


Figure 2: The basic RMM diagram for the JMIS Website.

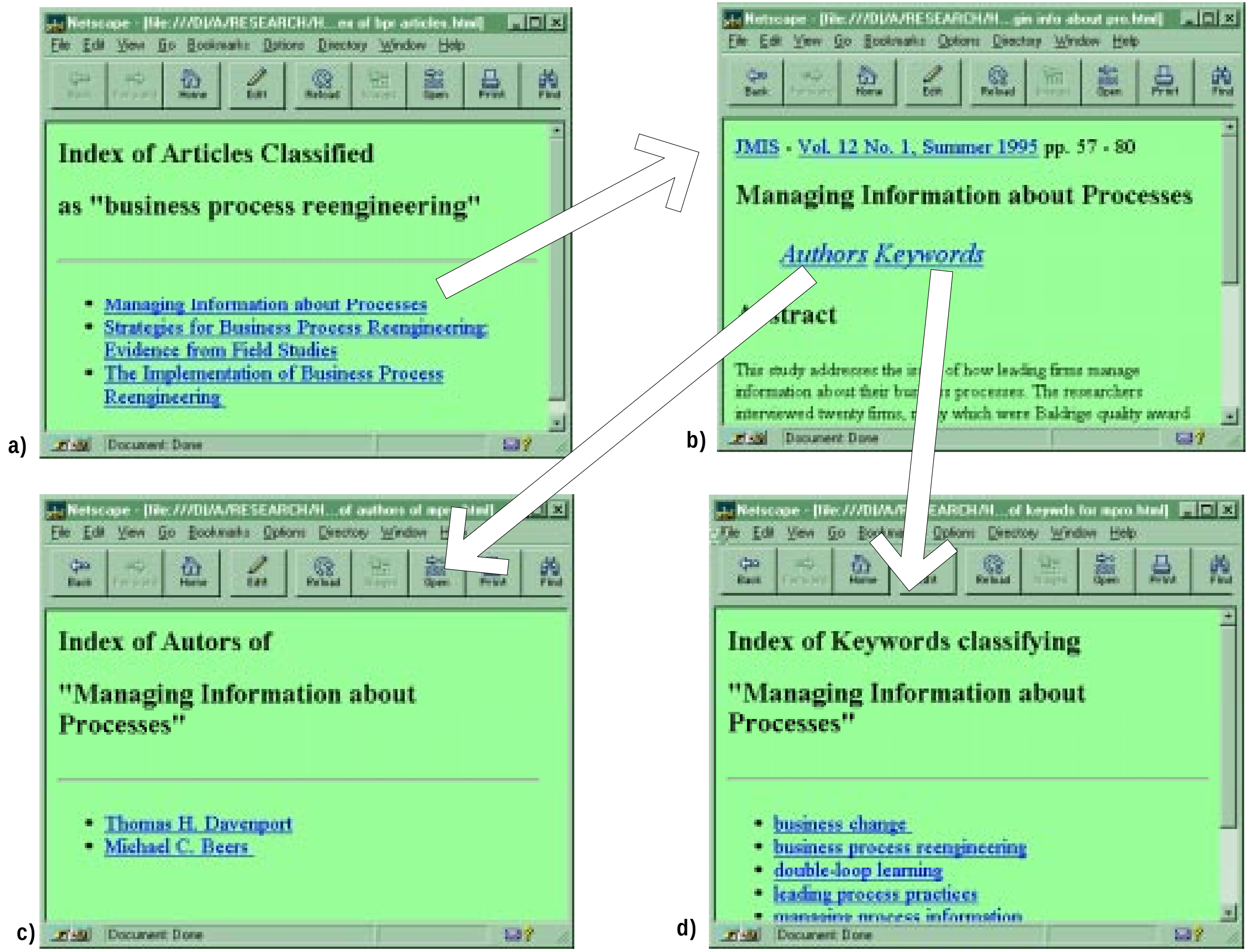
The RMM diagram of the JMIS Website designed with the current methodology is shown in Figure 2. One typically starts at the site's homepage, "JMIS Homepage", which is a *grouping* in RMM. From there, one can click on the keyword index to obtain a list of all keywords. Clicking on one of them would show all the articles classified by it. One can then select an article and click on its title to bring up the article page. From there, one can select the first author and click on the name to see information about him or her.

Figure 2 also shows a derived (inferred) relationship between the entities **article** and **journal**. Such relationships are used in accordance with HDM (Garzotto, Paolini, and Schwabe, 1993) and help to make the design of a hypermedia application more straight-forward. By clearly showing the derived relationships on the diagram, one does not need to explicitly trace them back to the composition of the actual relationships. In this example the derived relationship **in_journal** between **article** and **journal** is the composition of the existing relationships **in_issue** and **in_journal** between the entities **article** and **issue** and the entities **issue** and **journal** respectively.

3. Limitations of RMM

In the final implementation of the Website we realized that RMM overly constrains the resulting application. Suppose, for example, that for each keyword, we would like to produce a list of all articles classified by it and describe each article with a full bibliographic reference, as illustrated in Figure 4.

An RMM-driven implementation of the JMIS Website, based on the diagram in Figure 2, results in a number of separate pages, one for each construct shown in the diagram (assuming each entity has only one slice). The inability to put together, for example, the article name and the authors' names in the "Articles by Keyword" index, results in a cumbersome implementation. Figure 3 illustrates the navigational path that a user is forced to take in order to find the names of the contributors who authored a given article. Thus the user has to navigate two links in order to reach the screen in Figure 3-c, which at that point is devoid of context and meaning; the user may well have forgotten why he wanted that information. Compare this to Figure 4, which shows what can be accomplished by extending RMM as proposed in this article. The user can access the issue page, the article page, or the authors' pages with only one link.


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

a)

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

b)

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

Managing Information about Processes

Authors Keywords

stract

This study addresses the i of how leading firms manage information about their bu s processes. The researchers interviewed twenty firms, y which were Baldrige quality award

c)

Index of Autors of

"Managing Information about Processes"

- Thomas H. Davenport
- Michael C. Beers

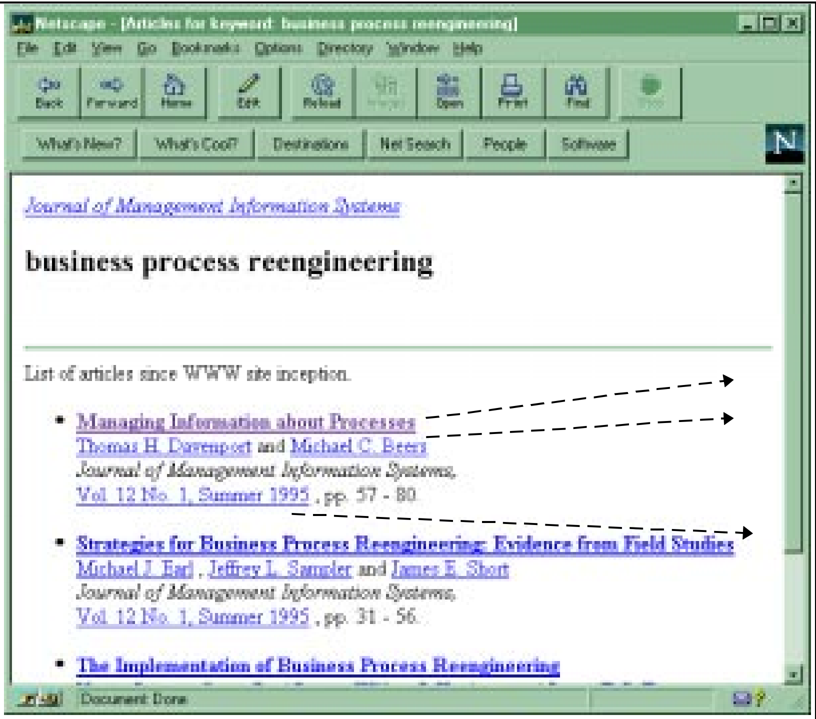
d)

Index of Keywords classifying

"Managing Information about Processes"

- business change
- business process reengineering
- double-loop learning
- leading process practices
- managing process information

Figure 3: Pure 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; and to d) in order to find other keywords classifying the same article. The extensions to RMM we present here enable richer designs that are more natural to the user.


Netscape - [Articles for keyword: business process reengineering]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
What's New? What's Cool? Destinations Net Search People Software
N
Document: Done

Journal of Management Information Systems

business process reengineering

List of articles since WWW site inception.

- 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.
- **Strategies for Business Process Reengineering: Evidence from Field Studies**
  Michael J. Earl , Jeffrey L. Sampler and James E. Short
  Journal of Management Information Systems,
  Vol. 12 No. 1, Summer 1995 , pp. 31 - 56.
- **The Implementation of Business Process Reengineering**

Figure 4: The use of m-slices provides for better navigation.

Based on our experience with the JMIS Website, we identified three main limitations in RMM:

1. The first limitation of RMM is its inability to specify what information is to be shown as the content of an anchor, i.e. the source for the actual text or image that appears as a hyperlink in a presentation unit such as a web page. For example, the title of an article is the content of an anchor that links the table of contents of an issue to that articles abstract page (see Figure 1)

2. RMM currently allows aggregation of attributes only within a single entity. This problem is solved by the introduction of a new kind of slice: the Matrjeska[1] slice (m-slice), which allows the aggregation of attributes from different slices. For example, in Figure 4, several slices are aggregated for the first article: "Managing Information about Processes" taken from the article, "Journal of Management Information Systems" taken from the journal, and "Vol. 12 No. 1, Summer 1995" taken from the issue. In our experience, we have come across m-slices that have five or more levels of nesting.
3. The third limitation is that RMM does not currently allow a slice to contain both attributes and access structures (e.g., indices or guided tours). To arrive at an access structure, one would have to traverse an extra link. The m-slice allows one to combine any slice with an access structure, without the extra link. For example, in RMM, "Thomas H. Davenport and Michael C. Beers" is represented by an access structure (an index). That index is combined with other information pertaining to the first article (see Figure 1).

It is important to note that the navigation afforded by RMM is the same in both versions. The difference lies in the ability to cluster together different elements of the application. This clustering ability is achieved through the use of the m-slice. The next section gives a detailed description of this new construct and shows how it can be used in the specific example of the JMIS Website.

---

[1] This is named after the Russian dolls that appear to be solid, but are in fact nested.

4. The M-Slice Solution

The construct we introduce here is the *m-slice*. M-slices are used to group information into meaningful information units. M-slices can be aggregated and nested to form higher level m-slices. The “m” in “m-slice” derives from the nested nature of Russian *Matrjeska* dolls. Ultimately, HTML pages on the WWW, for example, are the presentation units that visually render the higher level m-slices. Those m-slices may contain lower level m-slices that can be re-used a multiple number of times. Besides fostering reuse, this approach promotes structured design which is inherent in the definition of an m-slice.

M-slices are a new construct that replaces the slice and grouping constructs currently used in RMM. M-slice design preempts the original slice design and should be performed after the navigational design. In this paper we do not discuss design guidelines which we will undertake elsewhere.

Figure 5 depicts the rendering of the **article.bib_citation** m-slice, which contains bibliographic information about an article and relevant links.

contributon entit.y

attributes of related entities, and access structures such as indices. They can also be nested. For example, the **issue.date** m-slice, shown in Figure 6, which aggregates the **volume, number, season** and **year** of an issue, is included in the **article. bib_citation** m-slice.

We developed graphical and program specification languages to represent m-slices. The graphical language is to be used in a GUI tool to assist in hypermedia design. The specification language is to be generated by the GUI tool and read into a compiler or interpreter that associates data with m-slices to generate the HTML pages.

M-slices are a very powerful element of RMM. They allow a precise definition of information elements to be presented to the user while hiding (a) details - which are encapsulated in other m-slices, and (b) elements of the user-interface. Examples of (b) include the relative placement of the information elements on a screen, 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).

It is important to stress that m-slices describe *what* information is to be part of a construct and where to obtain it. M-slices do *not* dictate *how* this information is to be shown. That is left to the user-interface design stage of RMM. 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.

5. Graphical Notation for m-slices

The graphical notation for the design of the structural user interface of a hypermedia application uses many of the existing primitives of RMM as well as some new ones. A complete list of the primitives used is supplied in

---

[2] We also consider m-slices with no specific owner in section 5.

Appendix A and we describe many of them within the context of our specific example of the JMIS Web site. These graphical representations are immediately followed by the equivalent program code, which we discuss later.

The graphical notation for the m-slice, depicted in Figure 7, consists of the RMM entity and slice primitives which are enlarged and placed so that they partially overlap. The entity portion represents the owner entity of the m-slice. This means that the instance of this entity defines what information appears in this m-slice. Relationships within the m-slice use the owner entity as their source. The name of the owner entity is placed in the top left-hand corner of the rectangle.

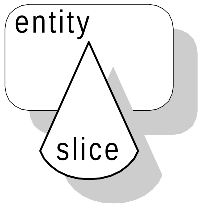


Figure 7: The m-slice primitive

The slice portion, whose name appears at the bottom of the drawing, contains the constituent elements of the m-slice. Note 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 (see Figure 8). The slice portion defines the physical boundaries. Elements appearing within those boundaries are the constituent elements of the m-slice. They are the ones that will physically appear in the presentation unit that is based on that m-slice. For example, everything that is visible on a web page is placed within the physical boundaries of the m-slice that the page represents.

There are, however, elements that are part of the m-slice's immediate external environment but do not appear 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 define the conceptual boundaries of the m-slice (Figure 8). If no such elements exist, then the physical and the conceptual boundaries of an m-slice are identical. Those 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 8.

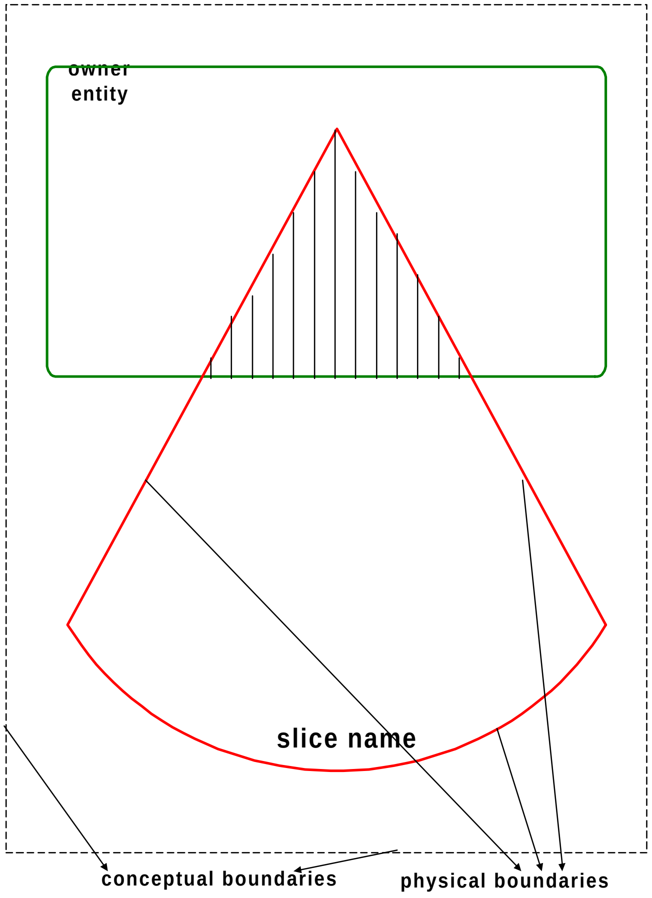
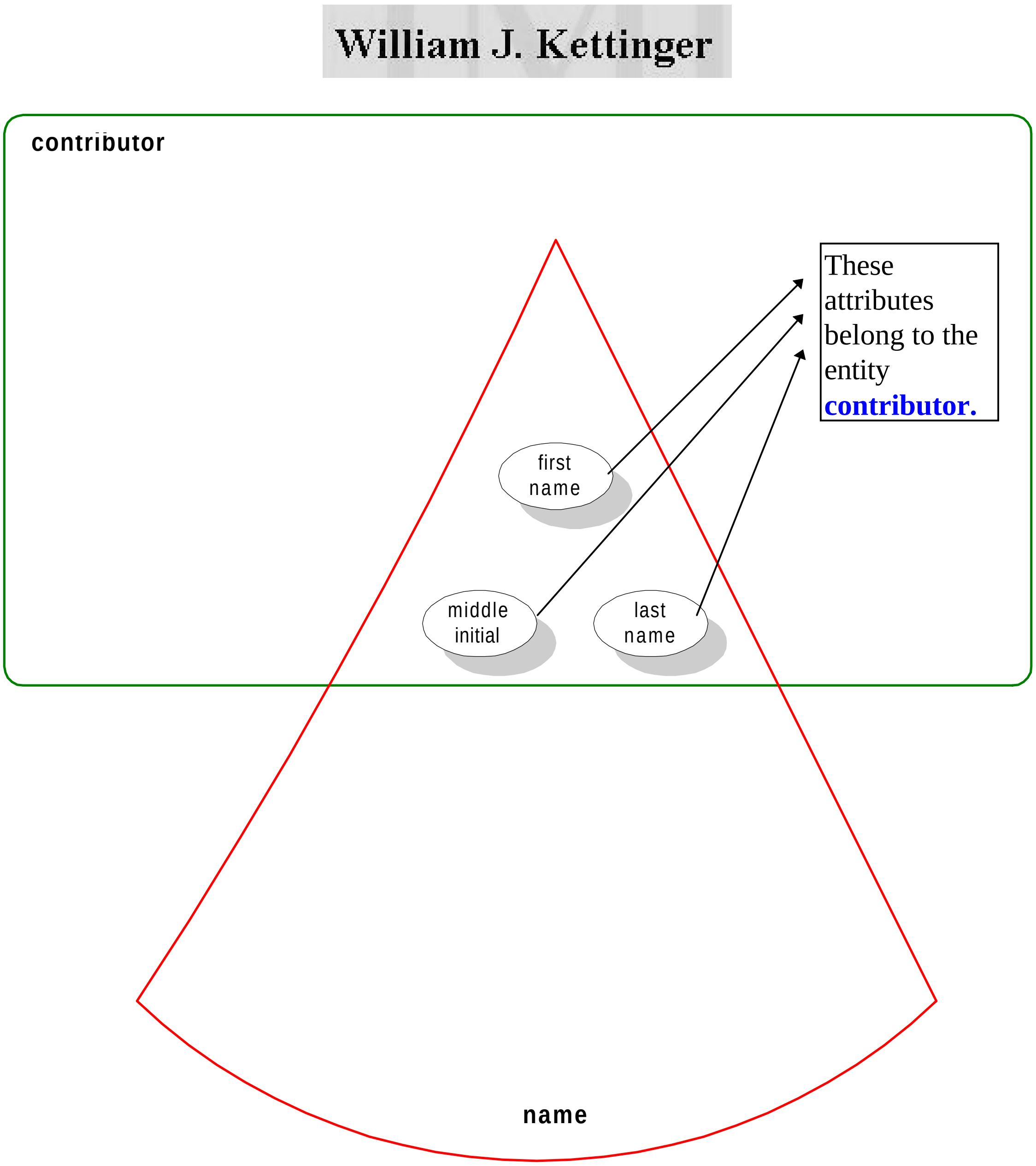


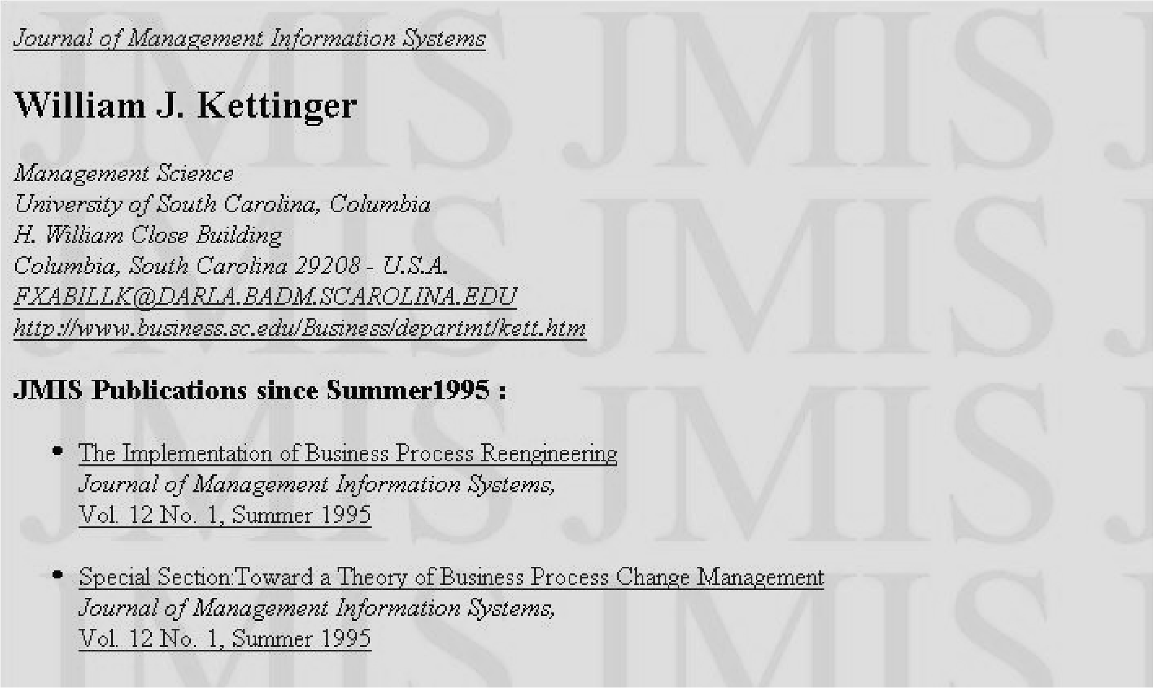
Figure 8: Physical and Conceptual Boundaries

Consider, as an example, the **contributor.name** m-slice (Figure 9). The three attributes **first name**, **middle initial**, and **last name** are attributes of the owner entity **contributor**. 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. Instead of an attribute, one can also use another m-slice belonging to the owner entity. For example, in **contributor.info** (Figure 10), the m-slice **contributor.name** is placed in the overlapping section to show that it is owned by the owner entity **contributor**. Note that no entity name is used in the description of the m-slice **contributor.name**.



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

Figure 9: The contributor.name m-slice



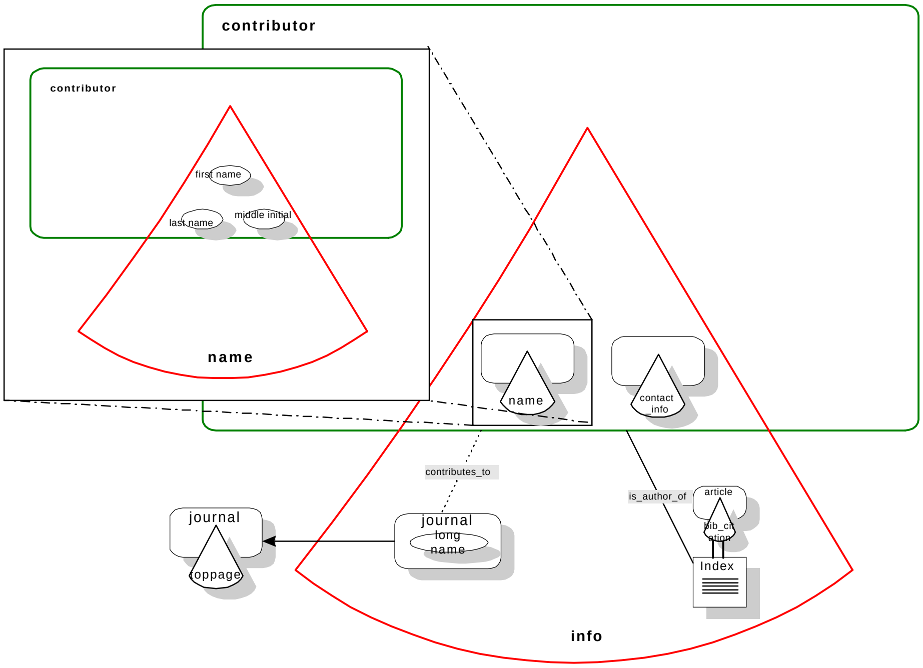
Journal of Management Information Systems

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

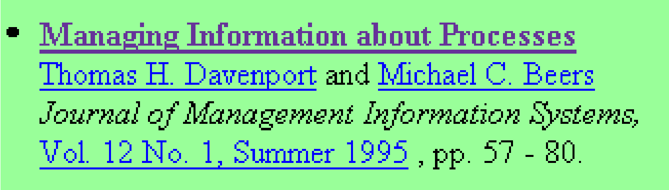


```
contributor.info: m-slice
begin
      [contributes_to]→ * journal.longname ⇒ journal.toppage;
      name;
      contact_info;
      index begin
            relation: [is_author_of];
            content: article.bib_citation;
      index end;
end;
```

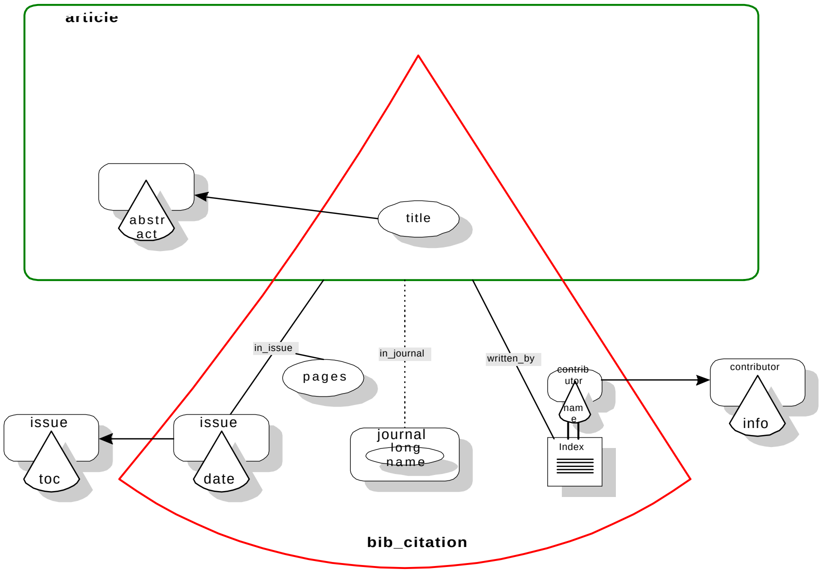
Figure 10: The contributor.info m-slice is an example of a nested m-slice (contributor.name).

This is also a good example of how m-slices can be nested to make the design of the hypermedia application more flexible. One can “explode” any nested m-slice to get its complete structure when using an on-line CASE tool.

An m-slice can also contain parts of entities 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 11 the m-slice **issue.date** is nested in **article.bib_citation.**



- **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;
      index begin
            relation: [written_by];
            content: * contributor.name ⇒ contributor.info;
      index end;
      [in_journal]→journal.longname;
      [in_issue] → * issue.date ⇒ issue.toc;
      [in_issue].pages;
end;
```

Figure 11: The article.bib_citation m-slice

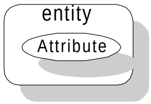


Figure 12: A single attribute m-slice

Since it is not part of the owner entity **article**, its primitive indicates both its owner entity, **issue**, as well as the slice's name, **date**. In many cases, m-slices contain just one attribute. To make their notation simple we use a shorthand notation, depicted in Figure 12.

Whenever we use an m-slice owned by another entity we must also specify the relationship between the two owner entities. Relationships are denoted by either a solid line for actual relationships or a dotted line for inferred relationships. The relationships always have the owner entity of the m-slice as their source, so the lines always start from the owner entity's border. An example of an actual relationship in **article.bib_citation** is **in_issue** while an example of an inferred relationship is **in_journal**.

M-slices can also combine different access structures such as indices. When an index is used, the relationship on which the index is based is indicated by the usual straight or dotted line. When designing the hypermedia application, however, it is necessary to indicate the information to be used as the content of the index in the m-slice. Therefore, two lines are used to connect the index with its content slice (Figure 13). Whether it's another m-slice or a single attribute, this information determines what will be used as the actual content of the m-slice based on the relationship that the index represents.

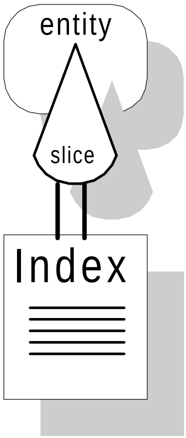


Figure 13: Index with content slice

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.

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**, seen in Figure 11**, pages** is an attribute of the relationship **in_issue** (between the entities **article** and **issue**). In cases such as this one, the attribute (or m-slice) is connected by a solid line to the relationship that owns it.

Sometimes, an m-slice is used as an anchor of a hyperlink to another m-slice. Such a link is shown by an arrow that crosses the border of the m-slice. In **article.bib_citation,** the m-slice **issue.date** serves as a hyperlink to the m-slice **issue.toc**. This is shown by a uni-directional arrow connecting the two m-slices while crossing the border of the m-slice **article.bib_citation**. The same is done for the index of the relationship **written_by** where its 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 link to an m-slice that is owned by the same owner entity. The attribute **title** of the entity **article** serves as a link to the m-slice **article.abstract**. Since that slice is also owned by **article** it appears outside the physical boundaries of the m-slice **article.bib_citation** but within the entity **article**.

A hyperlink can also initiate a process such as e-mail, a video, an audio file, or a file download. For all those cases, special primitives are used to indicate the external link or the process involved (see **Error! Reference source not found.**).

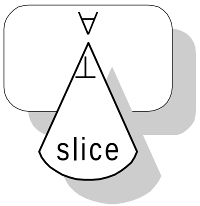
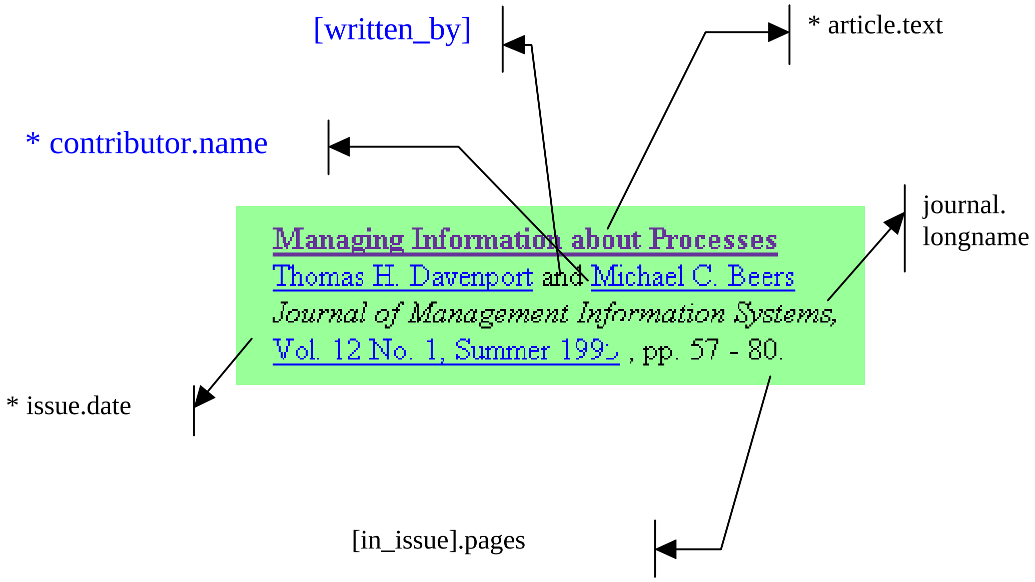


Figure 14: An empty slice

Often, we need to place a hyperlink in an m-slice that does not use information from the domain of the application. For example, we may want to use a fixed text or an image as a button that can be activated to open another page. We therefore need some primitive to serve as a placeholder. We call this placeholder an empty slice and its graphical notation is shown in Figure 14.

6. M-SLICE LANGUAGE DEFINITION

The merits of a program specification language are : (a) it provides precise definition and (b) it is executable. We illustrate the programming notation with several examples. A full description of the language is given in Appendix B. Let's build the **article.bib_citation** m-slice, starting with the simpler elements.



Managing Information about Processes
Thomas H. Davenport and Michael C. Beers
Journal of Management Information Systems,
Vol. 12 No. 1, Summer 199 , pp. 57 - 80.

Figure 15: Rendering of the bib_citation m-slice

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 relation, **in_journal**. The syntax for this is the following:

`[relation] → <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

`[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, in this case, pp. *57-80*. **Pages** is an attribute of neither the article nor the issue, but of the relation between them. Syntactically, this is as follows:

`[relation].<m-slice>`

and, in this example, is

`[in_issue].pages`

Next, we describe the *hyperlink* construct. Consider, from Figure 5, 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

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

More generally, the syntax for hyperlinks is as follows:

```
[relation] → * <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;
```

“**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 usually 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
      relation: <relation name>
      content: <m-slice> | <hyperlink>
index end
```

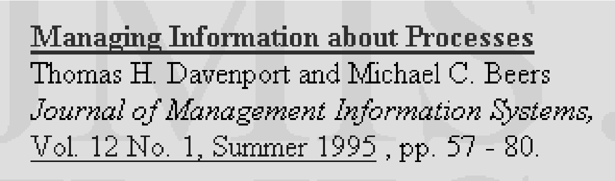
In this example, it is

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

The **article.bib_citation** m-slice would appear as follows:



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.

Figure 16: article.bib_citation without hyperlink

and it would not be possible to click on the contributors.

Putting all the pieces together, we have

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

Due to space constraints, the remaining constructs are described in Appendix B.

7. Related Work

The research we presented here is based on the original RMM methodology introduced by Isakowitz, Stohr and P. Balasubramanian (1995), which describes a simpler approach to the design of slices. However, as noted earlier, this approach is limited in many respects, particularly the ability to produce usable screens. More complex kinds of RMM slices were presented by V. Balasubramanian, Bieber and Isakowitz (1996), who describe the concepts of minimal and hybrid slices. Minimal slices act as default anchors for links, thus they are a predecessor of the hyperlink construct presented here. Hybrid slices aggregate elements from different RMM elements, in the spirit of m-slices, but cannot be nested. None of these articles, however, introduces the graphical and programming languages we discuss here.

Garzotto, Paolini, and Schwabe's HDM data model (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 (1996) and Schwabe and Rossi (1993) have proposed hypermedia design methodologies based on the object-oriented paradigm. For database domains, RMM has the advantage of using tools such as E-R diagrams, with which designers are already familiar.

OOHDM incorporates some of the same functionality as RMM within an Object Oriented framework. The OOHDM concept of navigational class schema, presented in detail by Schawbe, Rossi and Barbosa (1996), is similar to the m-slices described here. Although OOHDM has a programming-like language to describe navigational class schemas, it lacks a graphical notation. A key difference is that while m-slices focus on 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.

8. SUMMARY

We described the RMM limitations we encountered in developing a web site (http://www.stern.nyu.edu/jmis). Specifically, these were the inability to define the content of anchors and the inability to cluster elements from different entities. These limitations led to the development of some powerful extensions to RMM.

M-slices are a very powerful element of RMM. They allow a precise definition of information elements to be presented to the user while hiding (a) details - which are encapsulated in other m-slices, and (b) elements of the user-interface. We devised a graphical notation and programming language to facilitate the design and implementation of M-slices. The extensions described in this article prove useful for the design of and arbitrarily complex hypermedia applications in a rigorous and structured fashion.

The extensions we provide here have been developed to be consistent with the RMM process and notation so that they can be seamlessly integrated with existing hypermedia design software tools such as RM-CASE (Diaz, Isakowitz, Maiorana and Galliberti, 1995). The m-slice enhancement to RMM can provide the necessary power and flexibility to RM-CASE necessary for the design of complex hypermedia applications. At the same time it ensures that the applications are well-structured and easy to maintain.

REFERENCES

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

Balasubramanian, V., Bieber, M.P. and Isakowitz T. (1996). “Systematic Hypermedia Design.” CRIS Working Paper series. Stern School of Business, NYU.

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

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

Diaz, A., Isakowitz, T., Maiorana V. and Gilabert G. (1995). RMCase: “A Tool To Design WWW Applications”. *World Wide Web Journal*, Vol. 1, No. 1, 1995, pp. 559-566.

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

Garzotto, F., Mainetti, L., & Paolini, P. (1996). “Navigation in Hypermedia Applications: Modelling and Semantics”. *Journal of Organizational Computing and Electronic Commerce,* Vol. 6, No. 3, 1996, pp. 211-238.

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

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

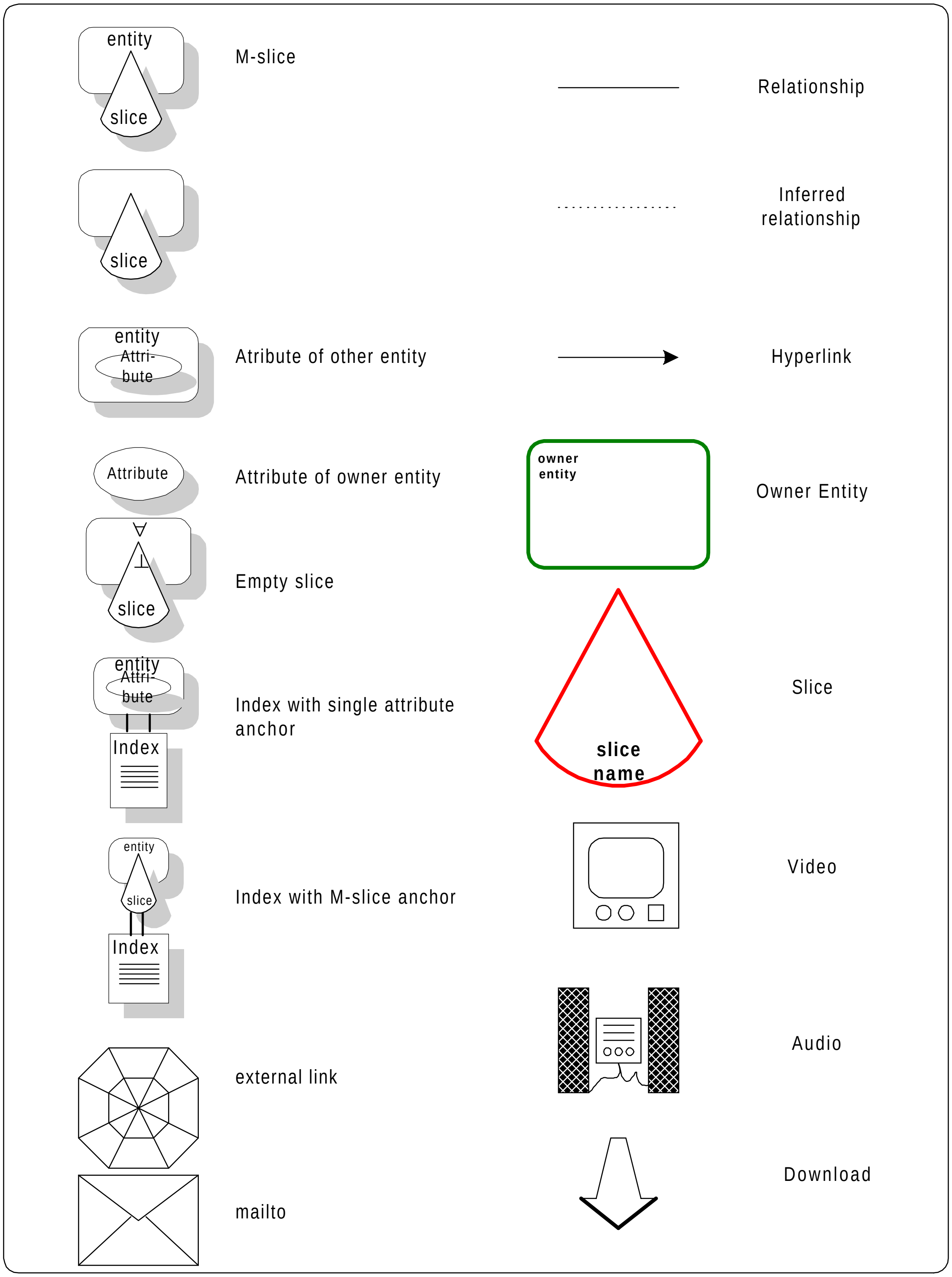
Lange, D. B. (1996). “An Object-Oriented Design Approach for Developing Hypermedia Information Systems”. *Journal of Organizational Computing and Electronic Commerce*, Vol. 6, No. 3, 1996, pp. 269-294.

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

Schwabe, D., & Rossi, G. (1995). “Building Hypermedia Applications as Navigational Views of Information Models” in Proc. *The 28th Annual Hawaii International Conference on System Sciences (HICSS ‘95).*

Schwabe, D., Rossi, G., & Barbosa, S.D.J. (1996). “Systematic Hypermedia Application Design with OOHDM” in Proc. *Hypertext '96*, 116-128.

Appendix A


entity
slice
M-slice
Relationship
slice
Inferred relationship
entity
Attri-bute
Atribute of other entity
Hyperlink
Attribute
Attribute of owner entity
owner entity
Owner Entity
∀
⊥
slice
Empty slice
entity
Attri-bute
Index
Index with single attribute anchor
slice name
Slice
entity
slice
Index
Index with M-slice anchor
Video
Audio
external link
Download
mailto

Appendix B

Complete Programming Language Specification for M-slices:

The meta-notation for an m-slice is the following:

<m-slice> :== *<owner entity>.<slice>*

where the slice *<slice>* belongs to the entity *<owner entity>*. For example, **article.abstract** refers to the slice **abstract** owned by the entity **article**. The notation for defining an m-slice is as follows:

```
<m-slice>: m-slice
begin
      <body>;
end;
```

For example, "**article.title**: m-slice" begins the definition of the slice **title** owned by entity **article**. The entity that owns the slice is called the owner entity. In the case above, **article** is the owner entity of the m-slice **article.title**.

The body of an m-slice is a set, possibly empty, of elements. Each element is one of

1. attribute or m-slice
2. hyperlink
3. access structure
4. empty slice

(1) ATTRIBUTE OR M-SLICE

An attribute or an m-slice of an entity, denoted as follows:

<attribute of an entity> :== *[relation]* → **<***entity>.<attribute>*

`<m-slice of an entity>` :== `[relation]` → `<m-slice>`

For example, the slice **name** owned by entity **journal** accessible from the defining entity via the relation [**in_journal**] would be denoted as follows:

`[in_journal]` → **`journal.name`**

When the attribute (or m-slice) is from the owner entity, we use **`this`** as the relation name, as follows:

`[this]` → **`article.title`**

To simplify the notation, we usually employ a shorthand, omitting the "*`[this]`*→*`<entity>`*" prefix. We thus can write

`title`

instead of

`[this]` → **`article.title`**

(2) HYPERLINK

A hyperlink consists of an *`<anchor>`* and a <*`destination`*>. The *`<anchor>`* is an m-slice that becomes the clickable area in the implementation (e.g., on the WWW). The <*`destination`*> is another m-slice. The notation is as follows:

`<hyperlink>` **`::=`** **`[`***`relation`***`]`** → **`*`** **`<`***`anchor`***`>`** ⇒ **`<`***`destination`***`>`**

where *`<anchor>`* and *`<destination>`* are m-slices and * ⇒ denotes a hyperlink. For example,

[**`in_journal`**] → * **`journal.name`** ⇒ **`journal.toppage`**

is a hyperlink whose clickable text is the journal name, and whose destination is the journal's top page.

(3) ACCESS STRUCTURE

An access structure related to the owner entity via a relationship defined in the RMM diagram. The index construct is as follows:

```
index begin
      relation ::= [relation]
      content ::= <m-slice> | <hyperlink>
index end
```

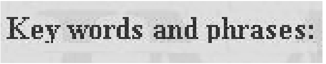
For example, the following is the index of contributors in an article:

```
index begin
      source: [written_by];
      content: * name ⇒ info;
index end;
```

The first part, `source`, defines the entity that provides the index contents and the relationship that determines which specific entity instances populate the index. In this case the source entity is **contributor** and the relationship is **written_by**. The second part, `content`, designates the elements (from the index source entity) to appear in the index.

(4) EMPTY SLICE

Consider the following index.



Key words and phrases:

This is from the **keyword.uses** m-slice. It is a placeholder that can be owned by any entity and it is defined as follows:

```
∀.keywords_and_phrases: m-slice
begin
end;
```

It is an empty slice. We use the bottom notation as a shorthand for it, as follows:

∀.**keywords_and_phrases-⊥**

Appendix C

The Relationship Management Methodology (RMM) in a Nutshell

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

Methodological Steps

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

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

The RMDM Data Model

The Relationship Management Data Model (RMDM) is the cornerstone of the RMM methodology. Figure 17 presents its elements. RMDM includes elements for representing information domain concepts (such as entities and relationships), and navigation mechanisms (such as links). An application's design is described via an RMDM diagram. The RMDM model is based on the Entity-Relationship model (Elmasri and Navathe, 1990), and on HDM (Garzotto, Paolini and Schwabe 1993) and HDM2 (Garzotto, Mainetti and Paolini 1996).

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

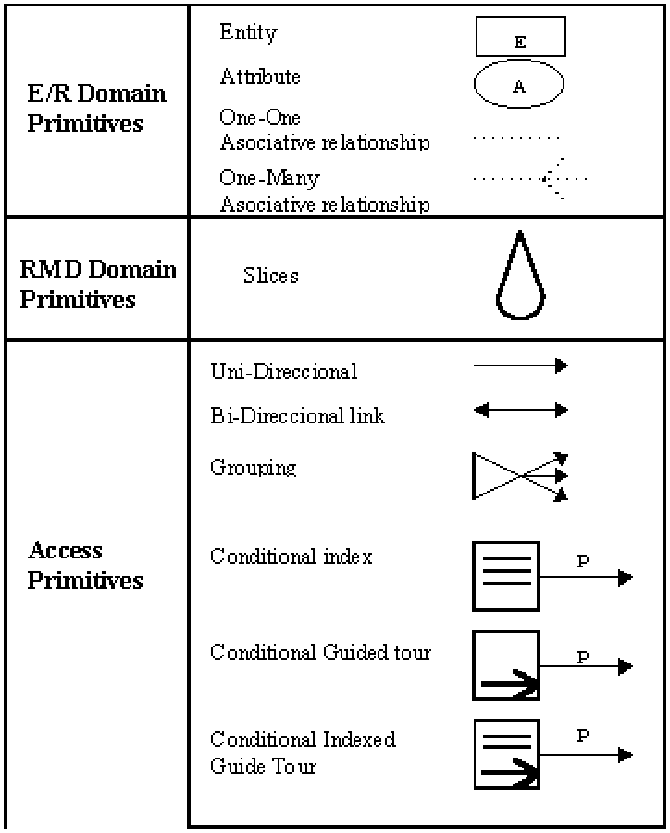


Figure 17: The elements of the RMM Data Model (RMDM)

RMDM specifies navigation via the six access primitives at the bottom of Figure 17. RMDM's most significant access structures are indices, guided tours, indexed guided tours and groupings. An index acts as a table of contents. A guided tour implements a linear path through a collection of items allowing the user to move forwards or backwards on the path. Indexed guided tours combine the functionality of indices and guided tours. Logical conditions qualify these access structures. For example, a condition "type=panel" attached to an index into a conference_event entity denotes an index to panels from that conference. The grouping mechanism serves as a major access gateway to other parts of the system, as often found on many applications' home pages or initial screens.