

# Extending the capabilities of RMM: Russian Dolls and Hypertext

Tomás Isakowitz  
Arnold Kamis  
Marios Koufaris

*Stern School of Business  
New York University  
44 W. 4<sup>th</sup> St.  
New York, NY 10012*

*tomas@stern.nyu.edu  
akamis@stern.nyu.edu  
mkoufari@stern.nyu.edu*

## 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*. The Relationship Management Methodology (RMM) [1] provides an effective, structured design methodology for the development of applications that are easier to maintain and extend. This paper assumes prior knowledge of RMM, a thorough description of which can be found in [1]. For a brief description, see [2]. Currently, RMM is in use at Merryll 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 displayed on each presentation unit, e.g. Web pages. When designing a website with 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 displayed within a single screen. Although RMM can currently model the basic navigational structure of a website it cannot combine various components in meaningful ways.

This paper extends RMM by introducing new constructs that overcome these limitations and increase its capabilities for the design of useful hypermedia applications. The graphical notation and program specification language that we present here for the first time can be used in an on-line case tool, such as RM-CASE [3]. While the original RMM methodology was limited to very simple applications, the extensions we provide here support the structured design of hypermedia applications of arbitrary levels of complexity.

In the paper, we first describe the website 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 information about each author. There is also a link to a keyword index that lists the available JMIS articles classified under each keyword. Throughout the site, references such as article titles, author names, and

its title to bring up the article page. From there, one can select an author's name to obtain information about him or her.

Figure 2 also shows a derived (inferred) relationship between the entities **article** and **journal** (shown with a dashed line). Such relationships are used in accordance

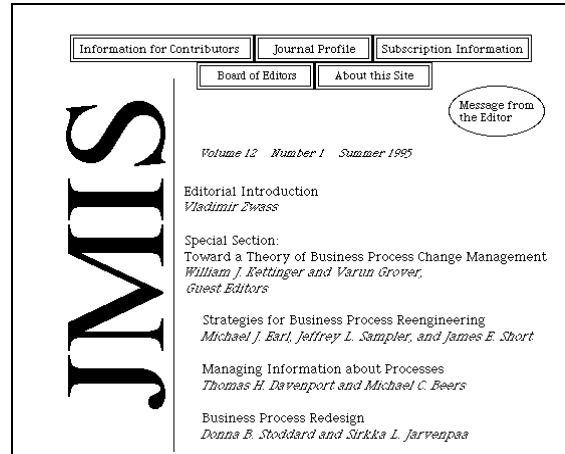


Figure 1: The issue table of contents page.

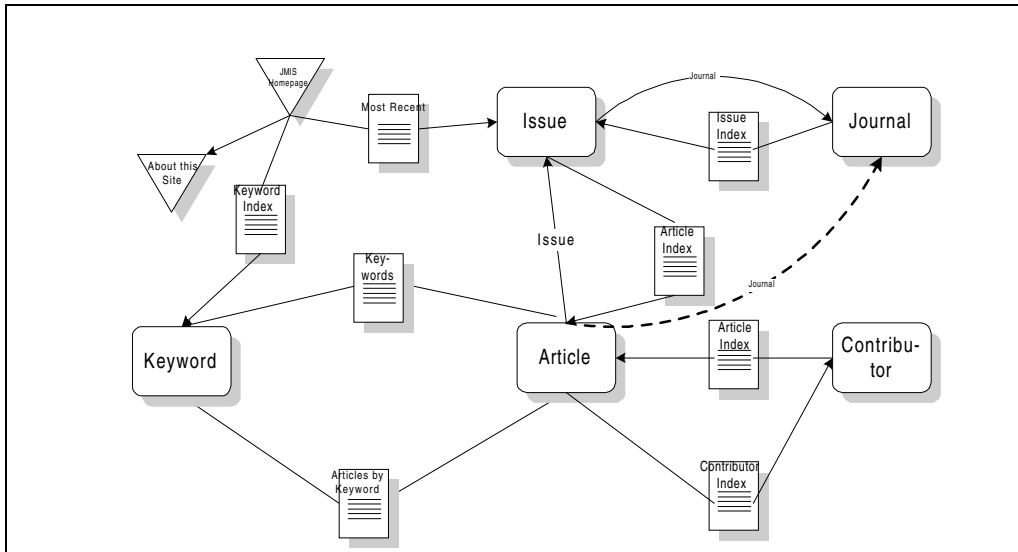


Figure 2: The basic RMM diagram for the JMIS website.

issue volume-number-season, are linked to the appropriate pages.

The RMM diagram of the JMIS website designed with the original 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 shows all the articles classified by it. One can then select an article and click on

with HDM [4] 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.

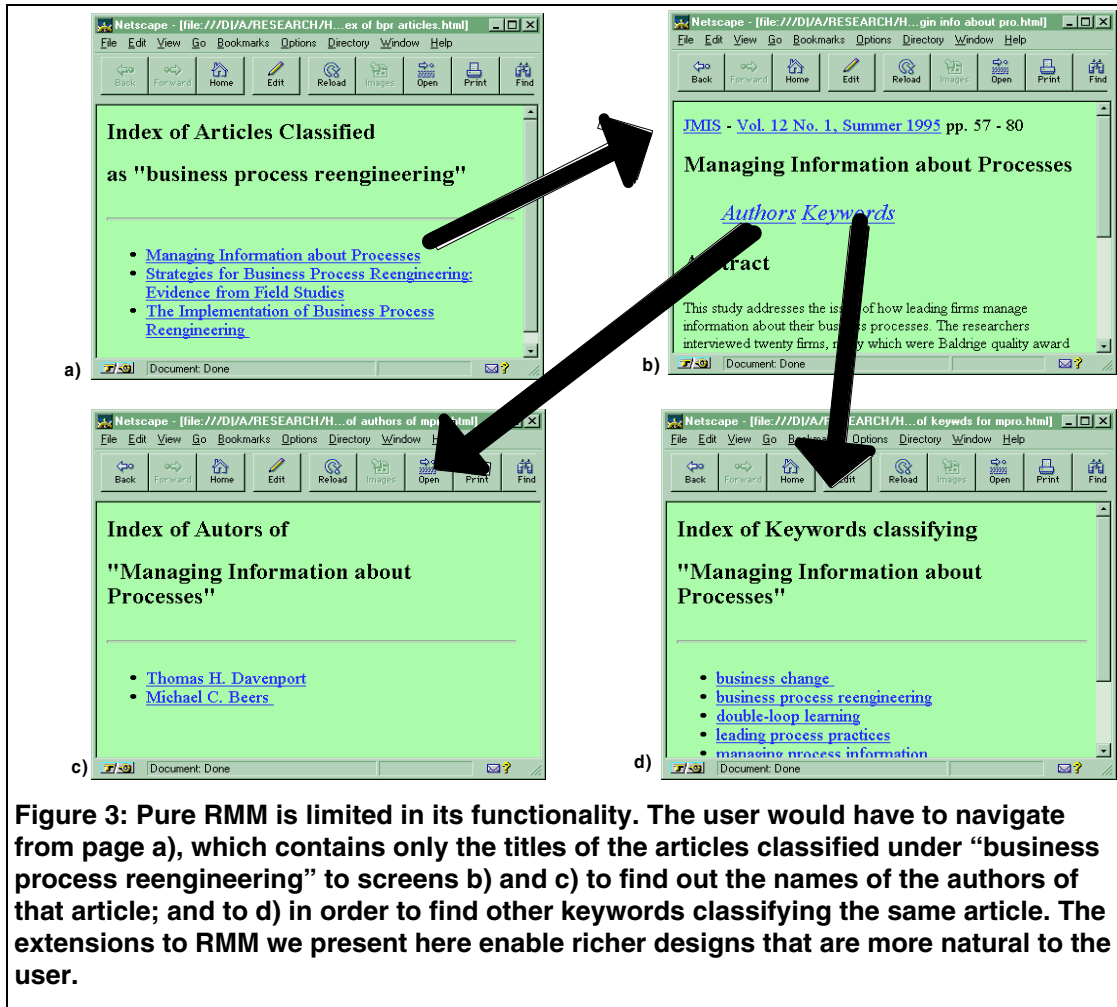
### 3. Limitations of RMM

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.

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 issue page, the article page, and the authors' pages are only one link away.

Based on our experience, we identified three main limitations in RMM:

1. The inability to specify what information is to be



**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.**

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 Figure 3-c, which at

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 article’s abstract page (see Figure 1).

2. Attributes can be aggregated only within a single entity. This problem is solved by the introduction of a new kind of slice: the 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 with five or more levels of nesting.



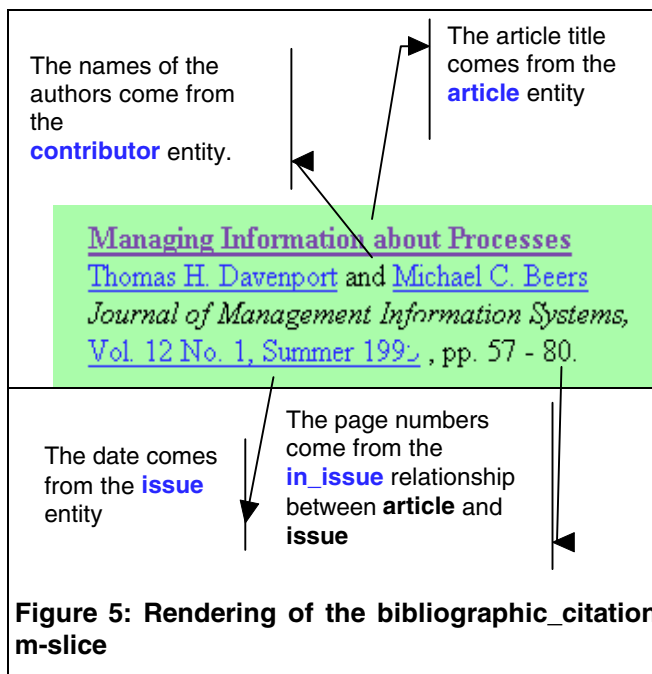
**Figure 4: The use of m-slices provides for better navigation.**

3. Slices cannot 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. M-slices allow arbitrary combinations without extra links.

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.

#### 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 *Matryeska* dolls. Ultimately, HTML pages on the WWW correspond to 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.



**Figure 5: Rendering of the bibliographic\_citation 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 in conjunction with the navigational design. In this paper we do not discuss design guidelines which we will undertake elsewhere.

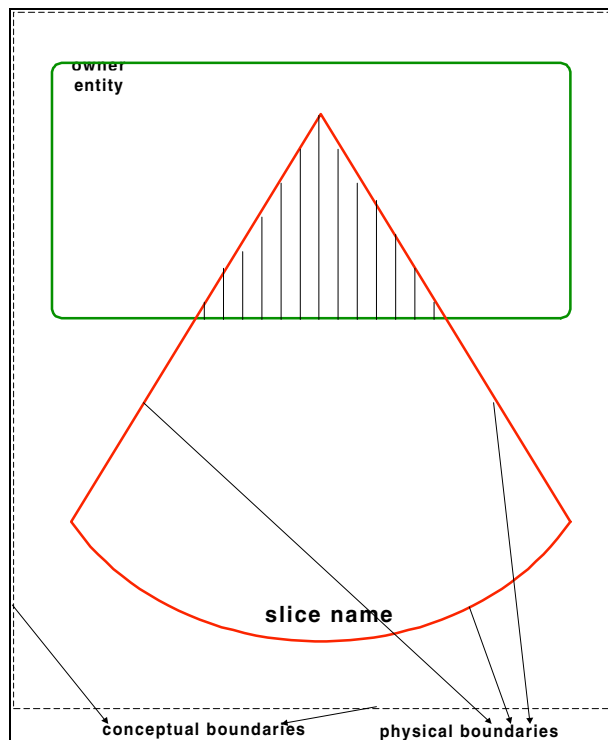
Figure 5 depicts the rendering of the `article.bib_citation` m-slice, which contains bibliographic information about an article and relevant links.

Each *m-slice* is *owned* by one specific entity<sup>1</sup> in order to be considered as an element of that entity. In that way, an m-slice can be reused as many times as needed, by itself or as part of another m-slice, without the need to redefine it. In the case of `article.bib_citation`, the owner is the `article` entity. To stress the role of owner entities, m-slices are denoted by `<owner entity>.<slice name>`. Note that in addition to containing 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 of the issue in which the article was published) comes from the relevant `issue` entity instance.

Thus m-slices encapsulate information from various sources: attributes of the owner entity, attributes of related entities, and access structures such as indices. They can

<sup>1</sup> We also consider m-slices with no specific owner in section 5.

also be nested. For example, the **issue.date** m-slice, shown in Figure 7, which aggregates the **volume**, **number**, **season** and **year** of an issue, is included in the **article.bib\_citation** m-slice.



**Figure 6: The M-slice Primitive, with Physical and Conceptual Boundaries**

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 HTML pages.

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 m-slices uses many of the existing primitives of RMM as well as some new ones. A

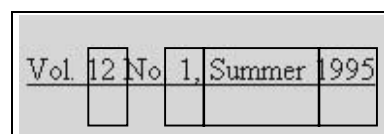
complete list of the primitives used is supplied in [5] and we describe many of them within the context of our specific example of the JMIS website. These graphical representations are immediately followed by the equivalent program code, which we discuss later.

The graphical notation for the m-slice, depicted in Figure 6, 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.

The slice portion, whose name appears at the bottom of the drawing, contains the constituent elements of the m-slice. 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 6. 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 within the physical boundaries of the m-slice is visible on a web page.

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 6). 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 6.

Consider, as an example, the **contributor.name** m-slice. 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 9), 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



**Figure 7: The issue.date m-slice**

m-slice **contributor.name**.

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 10 the m-slice **issue.date** is nested in **article.bib\_citation**. 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 13.

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 11). 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.

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 10, **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.

In many cases, 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** (Figure 10) we can also see an

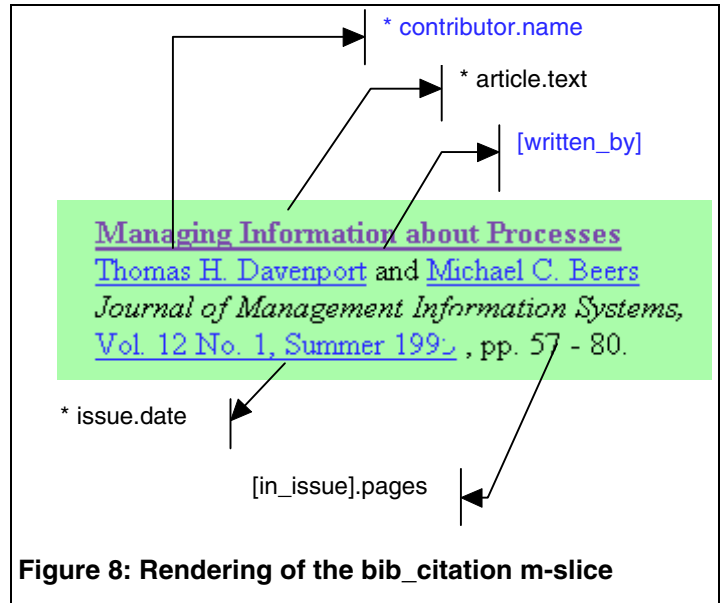


Figure 8: Rendering of the **bib\_citation** m-slice

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**.

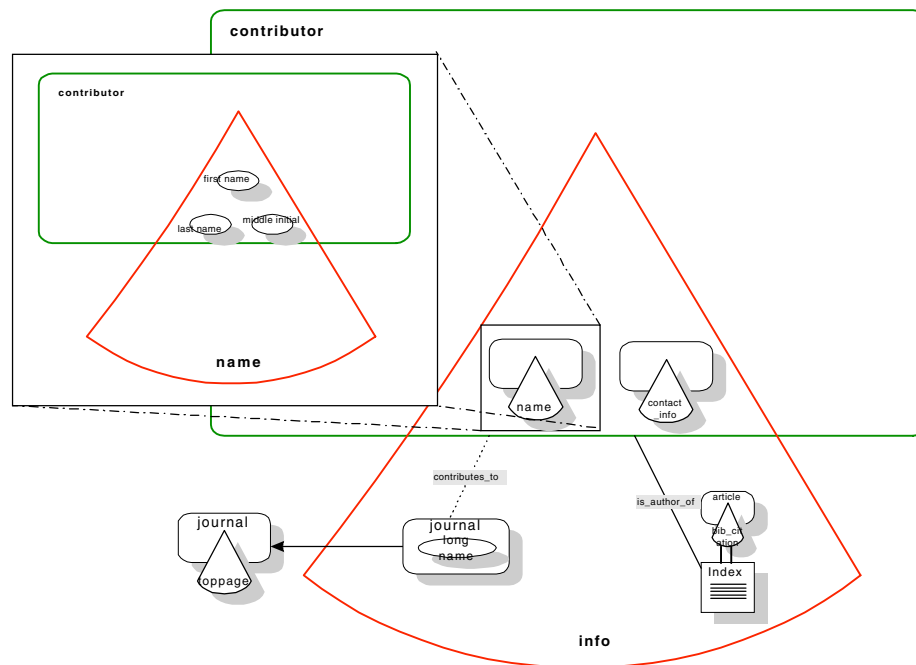
A hyperlink can also initiate a process such as e-mail, a video, an audio file, or a file download. For all those cases, special primitives are used to indicate the external link or the process [5].

## 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](mailto: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 9: The contributor.info m-slice is an example of a nested m-slice (contributor.name).

- [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.

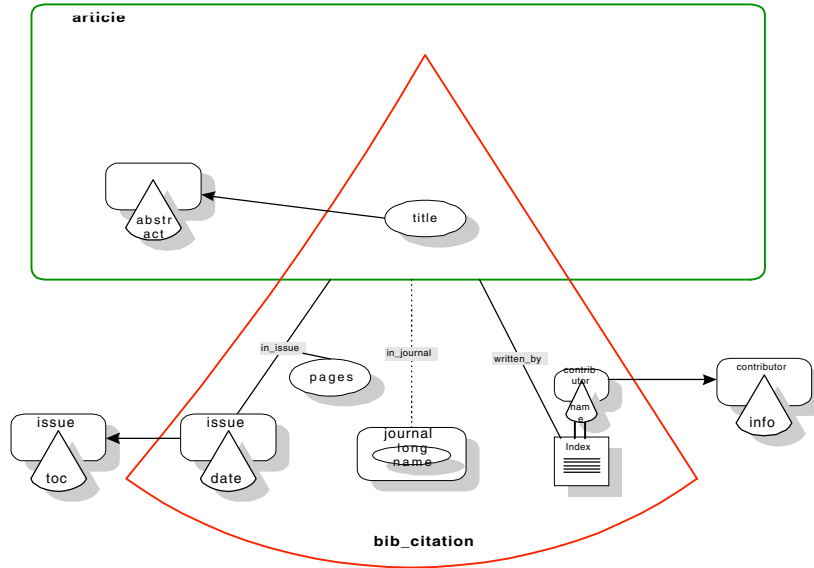


Figure 10: The article.bib\_citation m-slice

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 12.

## 6. M-slice language definition

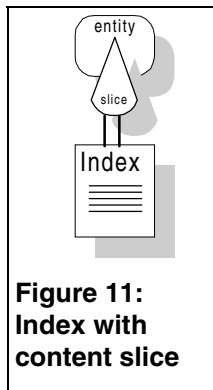


Figure 11: Index with content slice

```

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

```

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 [5]. Let's build the [article.bib\\_citation](#) m-slice, starting with the simpler elements.

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

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 as follows:

```

[relation] →
<entity>.<slice>

```

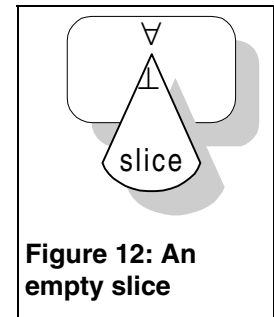


Figure 12: An empty slice

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, 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>

```

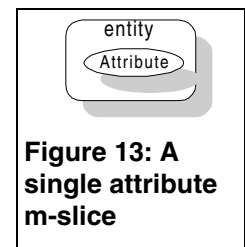


Figure 13: A single attribute m-slice



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

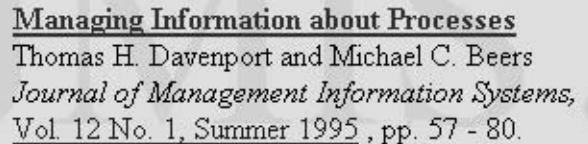
```
index begin
  relation: [written_by];
  content: * contributor.name ⇒ contributor.info;
index end;
```

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 14: article.bib\_citation without hyperlink.**

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

Putting all the pieces together, we have the following:

```
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;
```

For a full description of the programming constructs, we refer the reader to [5].

## 7. Related Work

The research we presented here is based on the original RMM methodology introduced by Isakowitz, Stohr and P. Balasubramanian [1], which describes a simple approach to the design of slices. However, as we noted, this approach is limited in many respects, particularly the ability to produce useful screens. More complex kinds of RMM slices were presented by V. Balasubramanian, Bieber and Isakowitz in [2], 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 [4] and its successor HDM2 [6] 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 [7] and [8] 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 the JMIS website. 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 [3]. 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

- [1] 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.
- [2] Balasubramanian, V., Bieber, M.P. and Isakowitz T. (1997). "Systematic Hypermedia Design." *International Journal of Human-Computer Studies*, forthcoming.
- [3] 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.
- [4] 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.
- [5] Isakowitz, T., Kamis, A., Koufaris, M. (1996). "Extending RMM". CRIS Working Paper series # IS-96-8. Stern School of Business, NYU.
- [6] 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.
- [7] 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.
- [8] 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)*.