

Ubiquitous Metainformation and the WYWWYWI Principle

Joseph Catanio*

JosephCatanio@att.net

<http://home.att.net/~josephcatanio/>

Nkechi Nnadi*

nn5@njit.edu

Li Zhang**

lxz9848@oak.njit.edu

Michael Bieber*

bieber@oak.njit.edu

<http://web.njit.edu/~bieber>

Roberto Galnares*

rxg3564@njit.edu

* Information Systems Department

College of Computing Sciences

New Jersey Institute of Technology

** Computer Science Department

College of Computing Sciences

New Jersey Institute of Technology

ABSTRACT

Computer systems should provide what you want, when you want it (the WYWWYWI Principle, pronounced “why why why”), but they frequently do not. Our research encourages a new philosophy of design based on the WYWWYWI principle, and the tools for authors to provide this easily.

Comprehensive metainformation embodies the WYWWYWI principle. Metainformation includes the structural relationships, content-based relationships, user-declared link-based relationships, and metadata around an element of interest. Combined, the metainformation goes a long way towards establishing the full semantics for (the meaning of and context around) a system’s elements.

We take a three-pronged approach to providing metainformation on a grand scale. First, we provide a systematic methodology for systems analysts to determine the relationships around elements of interest in their information domains—Relationship Analysis. Relationship Analysis will result in a comprehensive set of a domain’s structural relationships. Second, we provide a Metainformation Engine, which automatically generates sets of structural and content-based relationships around elements of interest as links, as well as metadata within static and virtual documents. Third, we provide an infrastructure for widespread link-based services within both static and virtual documents. This approach provides the inspiration as well as a sound foundation for a ubiquitous embracing of the WYWWYWI principle in the everyday systems people use, both on the Web and beyond.

1. MOTIVATION: WHAT YOU WANT, WHEN YOU WANT IT

How often have you wanted to point to something and say “tell me more about this.” People naturally want information on-demand. Computer systems should provide what you want, when you want it (the WYWWYWI Principle, pronounced “why why why” [Bhargava et al. 1988]). But they often do not. In on-line fiction this is not necessarily bad, but provides artistic tension. In adaptive tutoring systems the authors may feel the user is not ready for exposure to advanced information yet. With national security or competitive business sites, compelling reasons may

exist not to give access. Sometimes information simply is not available, too expensive to provide, or in an inaccessible format. But in the vast majority of cases, objects are not linked at all, or not linked to what we at a particular moment want due to lack of custom, lack of vision, incomplete analysis, neglect, by default, or even by design. Designers are not trained to provide maximum access, and there are few examples of truly richly linked Web sites [Bieber & Vitali 1997, Bieber & Yoo 1999]. We need a new philosophy of design based on the WYWWYWI principle, and the tools for authors to provide this easily and without overwhelming the user with choices (the cognitive overload problem). This flexibility should be available in our Web systems, and to the extent possible, in everyday life. Mobile systems, embedded computer systems and augmented reality systems all could give us the ability to point to an element of interest on the street, in a building site or on our car's dashboard and ask, "tell me more about this." (We use the term "element of interest" instead of object, so our approach is not confused with object-oriented programming. Object orientation certain can be used to implement our vision, but is not a fundamental aspect of it.)

Approach

For the sake of this paper, we will assume that the information that people want is accessible somewhere on the Internet or through an intranet. Information may be fee-based or available behind firewalls. The WYWWYWI principle can be applied equally well to communities with restricted access. For this discussion we shall ignore the many issues around charging and general access in all senses of this term [Shneiderman 2000]. Instead we concentrate on the vision of providing a plethora of useful links to the existing "metainformation" [Galnares 2001] around elements of interest (and hope that our vision will encourage people to make more information available and accessible). Metainformation includes the structural relationships, content-based relationships, user-declared link-based relationships, and metadata around an element of interest.

Metainformation is useful for many reasons. It can reveal a complete picture for a particular element, helping the user understand its different aspects and the surrounding context more fully, so he or she can utilize it in a fully-informed manner. Providing broad metainformation in the form of links also gives the user primary control over navigation, allowing him or her to get directly to where he or she wants to go. A limited number of link anchors, each of which leading to a single destination limit exploration and constrain users to a limited set of options thought best by the designer. Designers, of course, sometimes are correct in their understanding of what a user needs at the time they design each page. But how many times have you wished you could get different information than what was presented, or directly to a different destination?

Instead, we believe that whenever the user clicks on an element of interest, a set of multiple links to the metainformation should be displayed, customized to the user's current activity, preferences and interface. This set of links typically needs to be generated dynamically or "just in time," and automatically. This is because many everyday computer application systems people use are computational—their documents or screens are generated dynamically in response to user queries. (Such applications include database, analytical, decision support, enterprise support and legacy systems, both on and off the Web.) These "virtual documents" are only created when the user requests information on them. Thus, for metainformation to be ubiquitous and practical, links to it must be generated "just in time," at the time that virtual documents appear.

We take a three-pronged approach to providing metainformation on a grand scale. First, we are developing a systematic methodology for systems analysts to determine the relationships

around elements of interest in their information domains—Relationship Analysis. Relationship Analysis will result in a comprehensive set of a domain’s structural relationships. Second, we are prototyping a Metainformation Engine (ME), which automatically generates sets of structural and content-based relationships around elements of interest as links, as well as metadata within static and virtual documents. Third, the ME will provide an infrastructure for widespread link-based services within both static and virtual documents. We believe that our approach provides the inspiration as well as a sound foundation for a ubiquitous embracing of the WYWWYWI principle in the everyday systems people use, both on the Web and beyond.

We begin in section 2 with several illustrations of WYWWYWI metainformation support. Section 3 explores the notion of metainformation further, focusing on the structural relationships inherent in every system, and distinguishing these from content-based relationships. In section 4 we discuss link-based services, and how they must be extended for systems generating virtual documents. Section 5 presents our Metainformation Engine infrastructure, and the concept of Systems Integration through Linking, which it enables. We describe how the ME implements structural and content-based linking, as well as link-based services for virtual documents. Section 6 discusses techniques for combating the cognitive overload that large amounts of metainformation could engender. In section 7 we present Relationship Analysis. Section 8 brings together all these aspects of providing metainformation support and envisions how we might achieve the WYWWYWI principle on a global scale. Section 9 follows with some closing thoughts.

2. EXAMPLES

In this section we provide several examples of providing multiple links for systems.

A Purchasing Legacy System

We start with the vendor requisition screen from our university’s legacy purchasing system mocked up within a Web browser (see Figure 1). The actual system has no links, and non-expert users often find it quite confusing. One wants to point to the requisition codes and find out what they mean, and how one would add more. One wants to point to the vendor name and find out more information. Every piece of text on this screen could have links to a great deal of metainformation. The Metainformation Engine (ME) would automatically add anchors under each element on the screen. Clicking on any anchor would automatically generate a list of relationships, such as those shown in Figure 1 for the vendor. Selecting any of these would send the appropriate command to a separate information system to generate the desired information. Each of these anchors is a structural relationship, as section 3.1 elaborates. Section 5.4 describes the generation process in more detail.

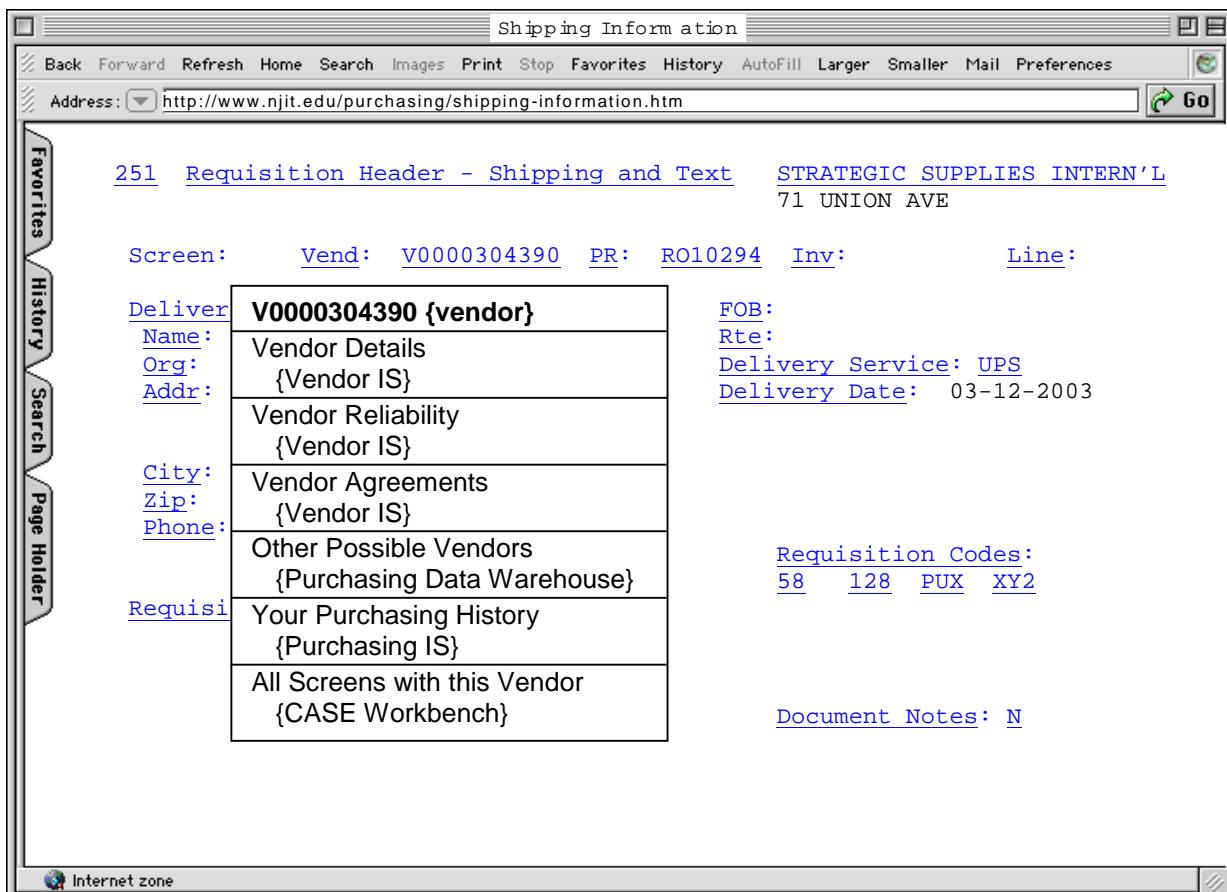


Figure 1: A mockup of a purchasing legacy system with metainformation support. The Metainformation Engine would consider almost every piece of text as a possible element of interest, and automatically add a link anchors under every one. Selecting the vendor identifier, for example, generates the set of 6 links to related information systems. Each link represents a structural relationship with the vendor element. Each link shows a display label describing the link, and the name of the destination system to which it leads.

Digital Library Collections and Services

Figure 2 presents a mock up for a digital library collection. As the document is displayed on the user's Web browser, the ME adds link anchors to the keywords it finds. It also adds a link anchor to the document as a whole. Figure 2 superimposes two possible sets of links. If the user clicks on the anchor that the ME added to the element of interest "Plant Pathology," the ME would generate its corresponding list of links with direct access to relationships and link-based services aspects in different collections and services relevant for this concept. If the user clicks on the document information icon (in the top right-hand corner), the ME would generate the corresponding list of links to seven relationships and link-based services relevant for this document as a whole. The ME also would filter and rank order the set of links generated to a specific user's preferences and current task. We currently are implementing this scenario in our Digital Library Service Integration project, and many of the systems mentioned in Figure 2's links are members of the National Science Digital Library [<http://www.nsdl.org>].

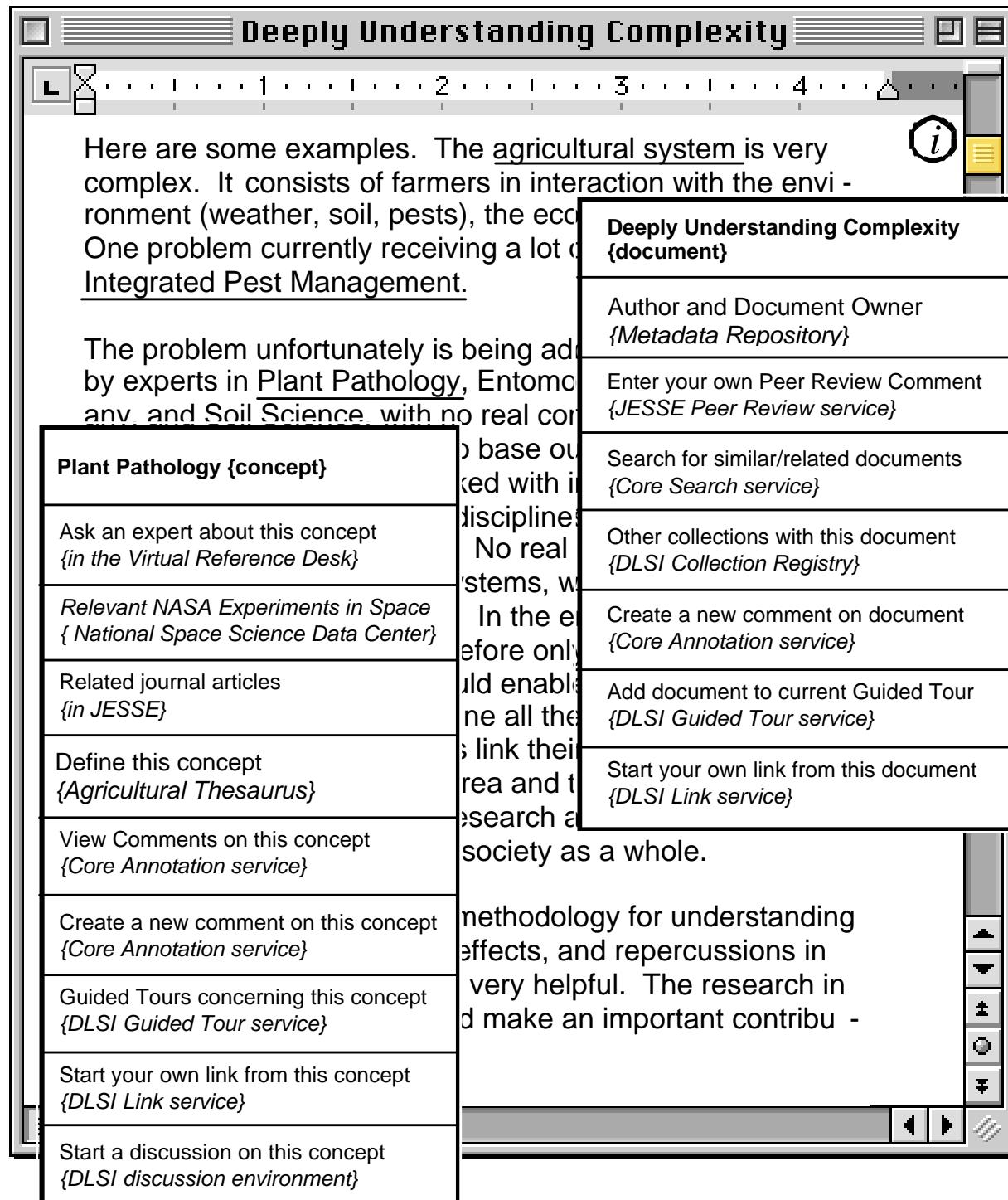


Figure 2: A mockup of a digital library document with metainformation support. The Metainformation Engine would automatically add link anchors, including an icon in the top right-hand corner for the document as a whole. Choosing one prompts the ME to generate a list of links. The figure superimposes two possible sets of links for different elements: the concept "Plant Pathology" and the document as a whole. Each link shows a display label describing the link, and the digital library service or collection it leads to.

Augmented Reality

Our third example builds upon some revolutionary work applying augmented reality to an architectural firm [Grønbæk et al. 2003]. The researchers added radio frequency identifier (RFID) tags to physical objects in the office, such as paper documents, binders, cardboard models, building materials and boxes. By pointing a small portable RFID reader at the RFID tag (or moving the tag in front of a reader), a digital representation of the object appears on a nearby computer screen. Why not now generate link anchors over the object itself, as well as to any elements of interest within the object? Clicking on one would generate a list of relevant links to related documents (such as blueprints), architectural elements, analysis routines and reference materials.

Similarly, functionality could be provided in augmented reality environments where people wear head mounted displays with glasses that superimpose an augmented computer screen over real-world objects. For example, as a visitor walks through specific sites in Athens, Greece, a bicycle helmet tracks his or her location and the ARCHEOGUIDE system renders a reconstruction of the original building or artifact over architectural ruins [Vlahakis et al. 2001]. Why not now generate link anchors over the building itself, as well as to any elements of interest within the rendering? Selecting one would generate a variety of audio or visual metainformation about that artifact, such as history, use, materials, links to related or contrasting artifacts, etc. Currently the ARCHEOGUIDE system provides some of this information in an audio recording automatically along with the rendering, but why not allow the user to select from a broad range of options?

As a last example, consider your car's dashboard and radio, where we could assume every indicator, switch and dial to be an element of interest, and a *de facto*, voice-activated link anchor. Mentioning its name (or some other identifier) could yield a set of relationships from the car itself. (For safety reasons, links to elements requiring concentration might only be supplied when the car is not in motion, and otherwise only to a passenger.) For example, the driver could ask about the gas (petrol) level indicator and gain access to the margin of error in the level shown, how many miles until the gas is used up, the current gas mileage, how much it would cost to tank up at the upcoming service station, how the engine uses gas, where gas comes from and how it is refined, how the level indicator works, and so on.

3. METAINFORMATION

Metainformation includes structural relationships, content-based relationships, user-declared link-based relationships, and metadata around an element of interest. Combined, the metainformation goes a long way towards establishing the full semantics for (the meaning of and context around) a given element of interest, such as a particular vendor, and thus implementing the WYWWYWI Principle. In this section we discuss structural relationships and content-based relationships, closing with a note on recent Semantic Web efforts, which could contribute to an element's metainformation. In section 4 we discuss related link-based services.

The notion of metainformation expands on what people typically consider metadata. Whereas metadata often describes characteristics of an element of interest, the other relationships often point to other entities or documents, as well as to functions that can be executed over aspects of that element.

In some of our earlier ME prototypes we experimented with displaying the metadata in a separate frame from the frame containing structural and content-based links, and link-based

services. When the user clicked on an element's link anchor, we would display the metadata for that element as well as generate a list of links to related items and services. However, we often found it difficult to decide whether to represent some information, such as a element's price or an annotation as metadata or a supplementary link. Relationship Analysis, for example, inherently yields both, with metadata often being described under the "characteristic" relationship.

We shall not describe metadata further, which is quite an active research topic in its own right. We shall explain how we generate it automatically, however, in section 5.

3.1 Structural Relationships

Structural relationships apply to an entire class of elements in an information domain. Structural relationships are inherent to the design or "structure" of the system. A database entity-relationship diagram, for example, contains structural links (though often just one per entity/element, and many fewer than WYWWYWI would encourage).

Structural links can connect the equivalent element, such as the same person or account. They also can connect related elements (such as employees working on the same project) or characteristics of an element (such as an employee and his or her address). The connected elements can be in the same or different systems. Thus a user may follow a link from an element in a specific system to the related element another in a completely different system. Often the destination system will need to execute a command (e.g., a query) to generate the destination element. This command will be associated with the link's relationship rule and executed if the user selects that link (see section 5.1).

In Figure 1 for example, the set of six links represents vendor relationships that apply to every vendor in the system. Figure 3 elaborates, showing the "semantic type" of the structural vendor relationships (and their classification within Relationship Analysis).

Similarly, in Figure 2, all of the links displayed (except those to link-based services—see section 4) represent structural relationships that would apply to any "concept" element and to every document, respectively.

In the majority of database, analytical, scientific, e-business and legacy systems, structural relationships can be produced automatically and on-demand. They often are result from some kind of retrieval or calculation commands. They often can be thought of as services, and indeed many structural relationship links could point to formal Web services. Web services are a technique allowing different systems to share independent modules that perform some kind of service (e.g., currency conversion).

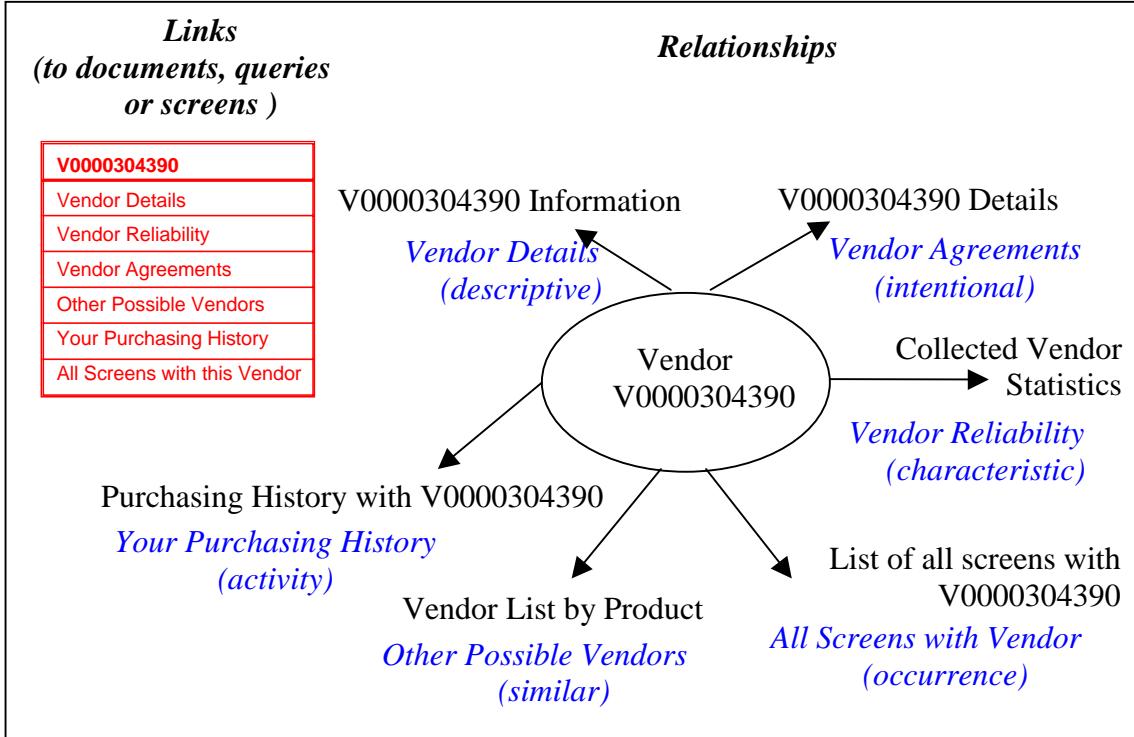


Figure 3: The correspondence between the links and their underlying structural relationships in Figure 1. The links point to the actual document, query result or screen produced by the destination system at the link's endpoint. The semantic types of the relationships and their classifications within the Relationship Analysis taxonomy are shown in italics. Together, with any additional content-based relationships, metadata and link-based services, these relationships help establish the full context around a given element of interest, such as a particular vendor.

3.2 Content-based Relationships

Content-based relationships also contribute to understanding the context around an element of interest. While equally important, they are not necessarily fixed in the structure of the related systems. Instead they are based on the display content. For textual content, content-based relationships typically are found using one or more of the many lexical analysis techniques ranging from simple keyword search to cluster analysis. In section 5.4 we describe the approach we currently use, but many others are possible.

Otherwise for multimedia content, much research is underway to automatically identify enough of the non-textual content to determine relationships. When alternate text tags provide metadata (such as the keywords on a photograph) or other identifying markup is available, then content-based relationships can be inferred for these surrogate representations.

Ontologies such as those being developed as part of the Semantic Web could be used to generate structural relationships among key terms. Part of understanding the context of an element could be to see how it fits within a web of related terms. A system supporting ontologies could easily be integrated within the ME architecture and add such links to the list that the ME generates. The ontologies also can be used in a more traditional way to find related terms for expanding keyword search and other content-based approaches.

4. SUPPLEMENTAL LINK-BASED SERVICES

Over the past 30 years or so, hypermedia, and more recently, World Wide Web researchers have developed a broad range of link-based services, which build on the concept of linking. Web developers have embraced some of these, but many others remain difficult to access, difficult to author, and difficult to share.

Link-based services include navigation, annotation and structural features [Bieber et al. 1997, Conklin 1987, Nielsen 1990]. These features serve many useful roles. They allow people to navigate along links more effectively. They provide structures and cues that help people find the information they want, while keeping them oriented within a complex network or “web” of inter-linked resources.

Several link-based services allow people to create new kinds of metainformation that helps customize the Web for themselves and for others, conveying some knowledge they have about an element of interest. These can be personal annotations by users to themselves as a reminder or organizing device. Alternatively, they can be devices for knowledge sharing with others. (Each feature can have access permissions specified, e.g., for being created, modified, deleted, linked to, and commented upon by an individual, work group or the general public.)

Such knowledge-sharing features include user-declared links, bookmarks (favorites), comments, overview maps, trails and guided tours. Overview maps provide a graphical view of a “web” of documents, often presented with the documents as icons and the links as arrows, often showing document names, and document and link semantic types [Cockburn & Jones 1996]. Global overview diagrams provide an overall picture; local overviews provide a fine-grained picture of document’s local neighborhood. Both provide spatial context and reduce disorientation (see section 6). Users can traverse directly to a document by selecting its icon.

Trails or paths connect a chain of links through an information space. They provide a context for viewing and understanding a series of documents. They can record a path of information to remember or share. They can suggest a subset or ordering within a “non-linear” web of documents, which can reduce cognitive overhead (see section 6). Authors can prepare multiple “recommended” trails, each focusing on a different aspect of a web or tailored to different readers (a novice, an expert, etc.) Guided tours are restricted trails with link anchors that lead away from the trail dimmed or hidden. Users have to suspend or exit the tour to access these. Trails can contain branches allowing the user to choose among sub-paths. Trails are often displayed and highlighted within an overview diagram, so users can maintain their orientation.

Hypermedia researchers strongly believe in the reader as author [Burton et al. 1995, Conklin 1987, Cotkin 1996, Miller 1995, Nielsen 1995]. All users should be able to add their own annotation and structural features, as well as additional links to any document. The World Wide Web has yet to fully embrace this vision, but it is a major facet of WYWWYWI. In part, this is due to the lack of authoring tools and ready facilities for everyday users to widely publish (share) these link-based constructs. Most browsers support neither. Some legal concerns also need to be worked out, such as the right to link to or from, or annotate someone else’s document. A final constraint is managing these features for the virtual documents produced by many of the business, scientific and other systems many people use, which is the topic of the next section.

4.1 Virtual Documents and Link-based Services

Many everyday computer systems are computational, their documents and screens are virtual, i.e., these are generated dynamically in response to user actions. Virtual documents only exist when the user visits them. They have no persistent state; upon leaving them they disappear [Watters & Shephard 1999]. Their contents are not necessarily stored anywhere.

Virtual documents have important ramifications for metainformation and link-based services [Zhang, 2004].

First, suppose the user created a link to an element within a virtual document or a bookmark to it while it was being displayed, and then closed the document so it no longer exists. When the user follows the bookmark or link, the virtual document must be *regenerated* at that moment.

Second, any anchors on elements that had been placed within that document previously, need to be *re-located*. If the elements' contents have changed, then they need to be *re-identified* so the anchors over them can be re-located. This includes any anchors to link-based services.

This is especially important when the same elements of interest appears in multiple virtual documents. Suppose the user places a link or annotation over a particular element of interest, such as an employee identifier, and also that the user specifies that this link or comment should be accessible every time that the element appears in any document. This is the issue of “anchor globality”. This means that individual elements of interest must be re-identified persistently across documents.

Re-identifying elements has the additional complication that the display text of the element can change over time (e.g., a stock price) or be customized to appear differently (e.g., the format of a date, or content within adaptive systems). This will require the notion of “sameness” (how much an element or document can change while still being considered the same instance as before).

Third, when a document is created as the result of user interaction, it is possible that that document had been generated in the past. Thus, we must try to re-identify all documents to see if they existed previously, and therefore may have had any link-based services declared over them. This requires virtual documents to be identified persistently across generations.

As a corollary, this means that many link-based services now have to be extended to work over virtual documents. Virtual document specifications must be referred to in links, overviews and guided tours. Overviews, and structural and content search must be able to work over a document “web” that has not yet been generated. Such functionalities must infer the potential of node and link existence, and node content, based on available specifications. Historical record does not suffice; users should see what *could* be generated by a target system.

The primary difficulty is that the virtual documents are produced in a system external to the metainformation and link-based services that supplement them. It is impractical to require every system to notify supplemental services of changes to documents, as often the systems have no need to track this information in the first place.

One difficulty with handling virtual documents produced by an external system is that the system adding system has

A static document that has been edited outside your metainformation and link-based service also has the re-identification problem. When your service opens it, the contents also may have changed. Simpler ways exist, however, to indicate that a static document has been edited. For

example, the editing timestamp changes [Davis, 1995]. On the other hand, every time a virtual document is generated, the date and time varies. In this situation, even though the file size does not change, one must re-identify whether this document is the same as one encountered previously.

There are several mechanisms for systems to track changes within static documents. Many document management and other systems support versioning, so the differences could be analyzed between versions. Dix [1995] uses dynamic pointers to track synchronous group editing. Dynamic pointers describe updates to various forms of location information such as text insertion points, selections, other block markers and various link anchor points. Multiple dynamic pointers are used to track different users' editing behavior for the same object within a document. If this versioning and editing information could be conveyed to the metainformation and link-based services, that may assist them in re-identification and relocation of elements and link anchors. But this would only help with static documents. Virtual documents are not necessarily edited, rather they can be generated differently at different times and under different conditions.

5. SYSTEMS INTEGRATION THROUGH LINKING & THE METAINFORMATION ENGINE

Our philosophy towards implementing WYWWYWI is not to produce any metainformation ourselves. Instead we wish to provide both an approach to facilitate others to provide metainformation, and a convincing vision for designers and developers to embrace this approach. While we present our own implementation strategy here and intend to open it up for free use by anyone, we also welcome alternatives that can achieve the same goal of ubiquitous metainformation and WYWWYWI access.

Our approach comprises three major aspects: the Metainformation Engine, link-based services and Relationship Analysis.

The Metainformation Engine (ME) implements "Systems Integration through Linking." Instead of reengineering systems to work with each other, this light-weight approach allows every system to function independently of the others with minimal or no modification. Integration occurs dynamically and virtually. When a system provides a document or screen to display, the ME intercepts it and adds link anchors over every relevant element of interest for that user. When the user selects a link anchor, the ME generates a list of links to relevant items and services. When the user selects a link, the ME sends the corresponding command to the target system to produce the information desired. Integration occurs in generating the set of relationships and as control passes from one system to another.

To integrate an application system, an analyst must write a wrapper, define relationship rules, and identify any relevant thesauri or glossaries. The ME manages the relationship rules, as we describe below.

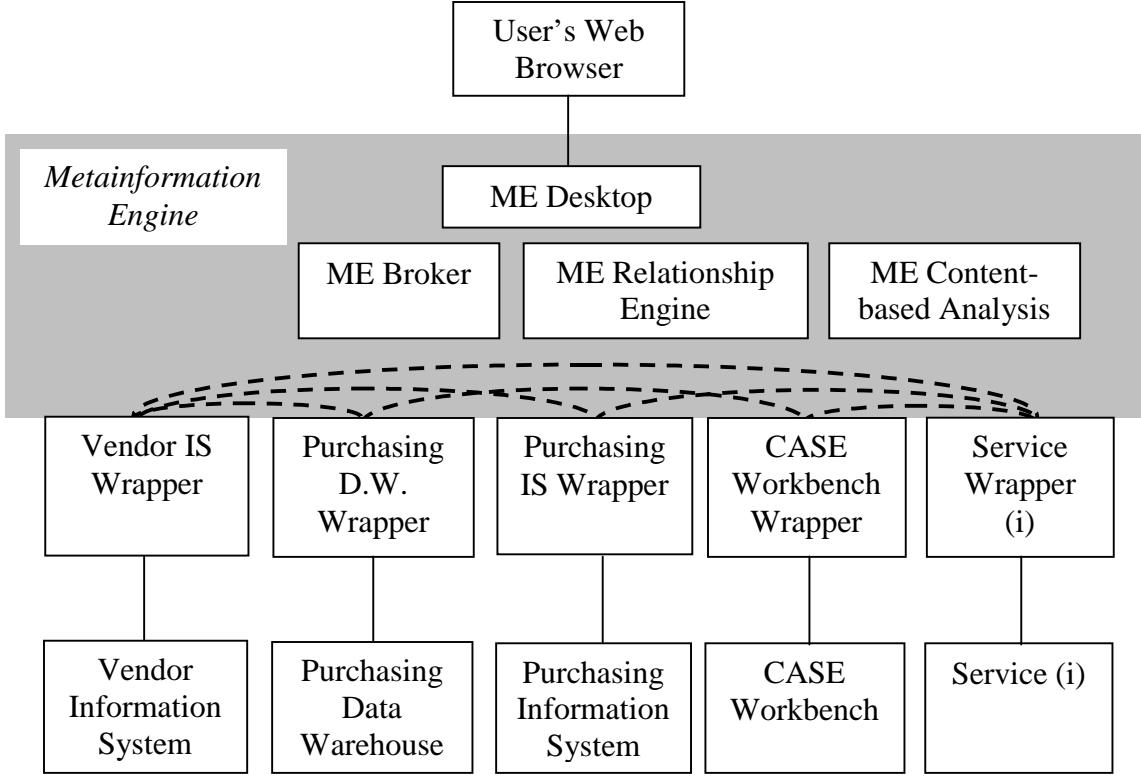


Figure 4: ME Integration Overview. The ME comprises the shaded area. The dashed paths indicate that once integrated, systems can share features through ME links automatically. Integrated systems also continue to operate independently of the ME. Section 5.4 describes the ME components further.

(1) *Develop a Wrapper:* The wrapper's main task is to parse the display screens that appear on the user's Web browser to identify the elements of interest that the ME will superimpose with link anchors. First, wrappers parse the display based on an understanding of the structure of its content. Second, the ME will parse the display content using content-based analysis to identify additional elements of interest. If any type of metainformation is available for a particular element, the ME will generate a link anchor for that element. Figure 4 provides a high-level illustration of wrappers connecting with the ME architecture.

(2) *Develop Relationship Rules:* We want to identify every service that each system can provide for any kind of element. Relationship Rules capture these services. Relationship rules specify the structural relationships and metadata relationships for recognized element types within the system being integrated. This defines the integration that occurs virtually along the dashed paths in Figure 4. When a user selects the link anchor over an element of interest, the ME determines many of the links to related information and services in every other system from the relationship rules that other system has declared. (The rest of the links are determined by content-based analysis.)

Relationship Analysis provides a systematic methodology for analyzing an information domain to determine its structural relationships and metadata. Integrators then can write relationship rules for each type relationship or metadata found during the analysis.

(3) *Identify Relevant Thesauri and Glossaries:* The ME's content-based (in this case, lexical) analysis refers to any available glossaries or thesauri to assist in identifying keywords and key phrases that have descriptions or any other information associated with them. When integrating a system, one should identify relevant glossaries and thesauri.

In the sub-sections that follow we shall describe relationship rules, integration of link-based services, content-based analysis and the ME architecture, including wrappers.

5.1 Structural Relationship Rules

Each structural relationship rule represents a single relationship for a single element class. As elements can have many relationships, each element class can have several relationship rules. Each element instance triggers the same set of relationship rules, assuming conditions are satisfied for each. For example, in Figure 1, the relationship rule underlying the first concept link would include the following parameters:

- a) the element type (in this case “vendor”)
- b) the link display label (“Vendor Details”)
- c) any relationship metadata (as opposed to element metadata, which has its own rules; relationship metadata describes the relationship or link and could include a link behavioral type (e.g., “query” or “computational”), semantic relationship type (e.g., “detail”), keywords, etc., which are useful for filtering links, and so forth) [Oinas-Kukkonen 1998]
- d) the destination target application system (the application programming interface (API) of the “Vendor Information System”)
- e) the exact command(s) to send to the destination system (e.g., “retrieve_full(ID, details)”)
- f) any relevant conditions for including this relationship (including the user types and tasks that would find this useful, level of expertise required, access restrictions, and so forth)

If the user clicks on this first link, the ME will instantiate the ID for this particular instance of vendor (“V0000304390”) and sends it to the destination Vendor Information System. The Vendor Information System will execute the command and produce a screen displaying the command results. Before this screen is displayed, the wrapper for the Vendor Information System will parse the screen contents, marking up each potential element of interest. The ME will add link anchors under each valid element so users can access sets of relevant relationships for this new screen. The WYWWYWI cycle thus continues indefinitely for all participating systems. Section 5.4 explains this cycle in detail.

The relationship rule underlying the fourth concept link in Figure 1 would include the following parameters, including a different destination system:

- a) the element type (“vendor”)
- b) the link display label (“Who Else Uses Vendor”)
- c) any relationship metadata (behavioral type = “command”, relationship semantic type = “other purchasers”)
- d) the destination system (the API of the “Purchasing Data Warehouse”)

- e) the exact command to send to the destination system (e.g., “vendor_users(ID, current_user)”, where the ME would instantiate the identifiers of this vendor and the current user when the user selects this link)
- f) any relevant conditions for including this relationship

In Figure 2, the relationship rule underlying the first concept link would include the following parameters:

- a) the element type (“concept”)
- b) the link display label (“Ask an expert about this concept”)
- c) relationship metadata
- d) the destination collection or service (“Virtual Reference Desk”)
- e) the exact command to send to the destination system (this would be a series of commands that automatically logs the user into the Virtual Reference Desk system, displays the URL of the question template, fills in the element instance (i.e., “Plant Pathology”) as the question subject, and places the cursor in the question area)
- f) any relevant conditions for including this relationship

To emphasize the core idea behind relationship rules: because they operate at the “class” or “kind of element” level, each relationship rule works for every element of that class or kind. This means that the rule just described applies to any “concept” found in any document displayed by any digital library collection or service, or indeed other application system. In Figure 2, nine relationship rules triggered for this “concept” element (or more rules triggered, but the filtering mechanism produced this customized list).

As we stated in section 3.1, many structural relationship links could point to formal Web services. One could specify a relationship rule providing a link to any relevant Web service (or group of similar Web services) appropriate to a particular class of elements within a system.

In our current ME implementation, relationship rules are stored in an XML database. A recent research project called *xlinkit* [<http://www.xlinkit.com>] is the only system we know doing something similar. They express relationship rules in first-order logic, which we actually did in an early prototype [Bieber & Kimbrough 1992, 1994], but have not yet re-implemented, instead concentrating on other functionality. In future versions of ME we hope to go back to this more flexible and powerful format, and will consider using *xlinkit* within an extended version.

5.2 Element Metadata Rules

Most element metadata is structural, i.e., the same parameters exist for each instance of an element class. The goal behind metadata rules is to represent all kinds of structural metadata for an element class, especially when parameters for the same element can be gathered from different systems.

For example, in Figure 1, one metadata rule underlying the vendor element would include the following parameters:

- a) the element type (in this example, “vendor”)
- b) the metadatum display type (“Vendor Name”)

- c) any metadata about this metadatum (semantic type (“name”), keywords, etc., for this metadatum itself)
- d) the system providing this metadatum (the application programming interface (API) of the “Vendor Information System”)
- e) the exact command(s) to send to the destination system (e.g., “select(vendor_table, vendor_ID, vendor_name)”)
- f) any relevant conditions for including this metadatum (including the user types and tasks that would find this useful, level of expertise required, access restrictions, and so forth)

5.3 Implementing Link-based Services

Link-based services can be provided through relationship rules. Whenever the user selects an element of interest, or a span of new text, the appropriate functionalities could be added. Figure 2 shows four link-based service links for the concept “Plant Pathology”:

- view comments on this concept
- create a new comment on this concept
- guided tours concerning this concept
- start your own link from this concept

and three link-based service links for the document itself:

- create a new comment on this document
- add this document to the current guided tour
- start your own link from this document

The following relationship rule could underlie the “view comments on this concept” service link. It would be valid for every type of element, including documents. A condition check would confirm whether any comments already exist for this element, in which case it would be included in the list of links.

- a) the element type (“generic_element”)
- b) the link display label (concatenate(“view comments on this”, element_type))
- c) any relationship metadata (behavioral type = “command”, semantic type = “annotation”)
- d) the destination system (“Core Annotation Service”)
- e) the exact command to send to the destination system (e.g., display_annotations(element_ID))
- f) any relevant conditions for including this function (check_condition(“Core Annotation Service”, existence_check(“annotations”, element_ID)) = true)

The following relationship rule could underlie the “add this document to the current guided tour” service link. A condition check would confirm whether the author is currently building a guided tour, in which case it would be included in the list of links.

- a) the element type (“generic_document”)

- b) the link display label (add this document to the current guided tour)
- c) any relationship metadata (behavioral type = “command”, semantic type = “add_to_tour”)
- d) the destination system (“DLSI Guided Tour Service”)
- e) the exact command to send to the destination system (e.g., retrieve_current_tour(tour_ID) and add_to_tour(tour_ID, element_ID))
- f) any relevant conditions for including this function (check_condition(“DLSI Guided Tour Service”, currently_building_tour = true))

5.4 Metainformation Engine Details and Related Research

This section begins by presenting the ME architecture. We then describe how we are augmenting this with just-in-time virtual document support (§5.4.2) and content-based analysis (§5.4.3). The final subsection presents related research (§5.4.4).

5.4.1 Metainformation Engine Architecture

The Metainformation Engine (ME) is a loosely coupled system, where various engine components communicate with each other via messages that conform to a well-defined standardized internal protocol. This approach allows new components to be developed and added without affecting existing engine components and functionality.

The ME’s goal is to supplement the output of most computer systems with link anchors and lists of links for each anchor, all with minimal or no changes. The ME will serve any application system and user interface (e.g., Web browsers) that has implemented an appropriate wrapper or “engine desktop”. To integrate a new system, it is necessary to develop a wrapper (i.e., identify elements, relationship rules, metadata, etc.) for it. Thus, to supplement the output of an system, the developer only has to develop and register the wrapper. This may prove straightforward or complex. But in any case it only must be done one time for a specific system to apply to any instance of that system.

Figure 4 shows an overview of the ME architecture. The ME consists of three primary components:

- The ME Desktop translates the displayable portion of the ME’s internal messages, from the standard internal XML format to a format that can be displayed to a user via a Web browser (or other kind of user interface) and vice versa.
- The ME Broker enables the communication between the Engine modules. All Engine messages pass through the Broker, which then redirects them to the appropriate engine component.
- The ME Relationship Engine (MERE) maps the system data and relationships to links at run-time. MERE maintains a repository of relationship and metadata rules. When a screen (or document) is being sent to the ME Desktop for display, MERE retrieves all relevant rules for each element in that screen. The ME Desktop then converts the elements to link anchors and the relationships to links.
- The ME Content-based Analysis performs lexical analysis as described in section 5.4.3. (In future research, we shall incorporate other forms of content analysis.)

All engine components that are capable of processing a ME message are known as enginelets. All system wrappers are enginelets. Enginelets can also be used to log intermediate messages, process for accuracy, provide filtering services, etc. These details are not shown to keep the discussion at a high level.

A system wrapper manages the communication between the ME and an application system, translates the user requests from the ME's internal format to a format the system can process, receives the output from the system and converts it to the ME format. The wrapper also parses, identifies and marks the elements of interest within the system's output.

A key characteristic of many (but not all) elements of interest is that their identifiers are not the same as their display content. For example, in Figure 1, while V00003040390 is the identifier of the vendor ID, it is also the identifier of the vendor name, which is displayed as "STRATEGIC SUPPLIES INTERN'L". A keyword search probably would not identify the vendor name. Similarly, a keyword search on the requisition codes most likely would turn up nothing meaningful in a keyword search. It is the job of the system wrapper to parse displays for the underlying elements they contain.

Coding a system wrapper is potentially the hardest part of system integration. If the system has an extensive application programming interface (API), then it usually is quite easy to parse the output displays (documents or screens) and detect the elements within it. If the system provides adequate metadata in tags or other ways (which is becoming increasingly prevalent with XML), parsing can take advantage of these and be straightforward. If documents/screens follow a well-defined template or format, which is the case with many e-commerce systems, then parsing also should be relatively easy. Otherwise, if a document or screen's content is unstructured and without embedded metadata, then the ME may have to rely solely on content-based analysis to identify elements of interest within it.

We shall explain the information flow with a digital library collection. Many collections have a well-defined document format, and thus we can write a wrapper to identify these structural elements. Alternatively, within the wrapper we could specify a template for each publication source within the collection (newspaper, journal, conference, etc.) that has its own consistent layout. Then if we know the source, we can then apply its predefined template.

Information flows through the ME as follows. Assume the user asked a digital library collection to display the document in Figure 2. The collection's retrieval function will pass the document to its wrapper. The collection's wrapper parses the document to identify possible elements of interest. First, the wrapper does a structural analysis; in this case for a research article it can easily identify the title, author, publication, sections, figures, etc. If the article included XML markup, further elements could be identified easily. Second, the wrapper uses a unified glossary of terms from participating collections and services to find keywords associated with the glossary entries. The wrapper forms an XML message in the ME's internal format containing the document and all elements identified, along with their object types, which it passes to the ME. The ME adds anchors for each element to a copy of the document, which is then passed to the user's Web browser for display. When the user selects any of these ME anchors, the ME Relationship Engine uses the relationship rules to generate a filtered list of links, which it passes back to the Web browser. (Figure 2 shows two sets of these links.) When the user selects one of these links, the appropriate set of commands associated with the relationship rules are passed to the associated collection or service. (For the first link in Figure 2's mockup, for example, the ME

Relationship Engine would use the relationship rule presented earlier to generate a query to the Virtual Reference Desk.)

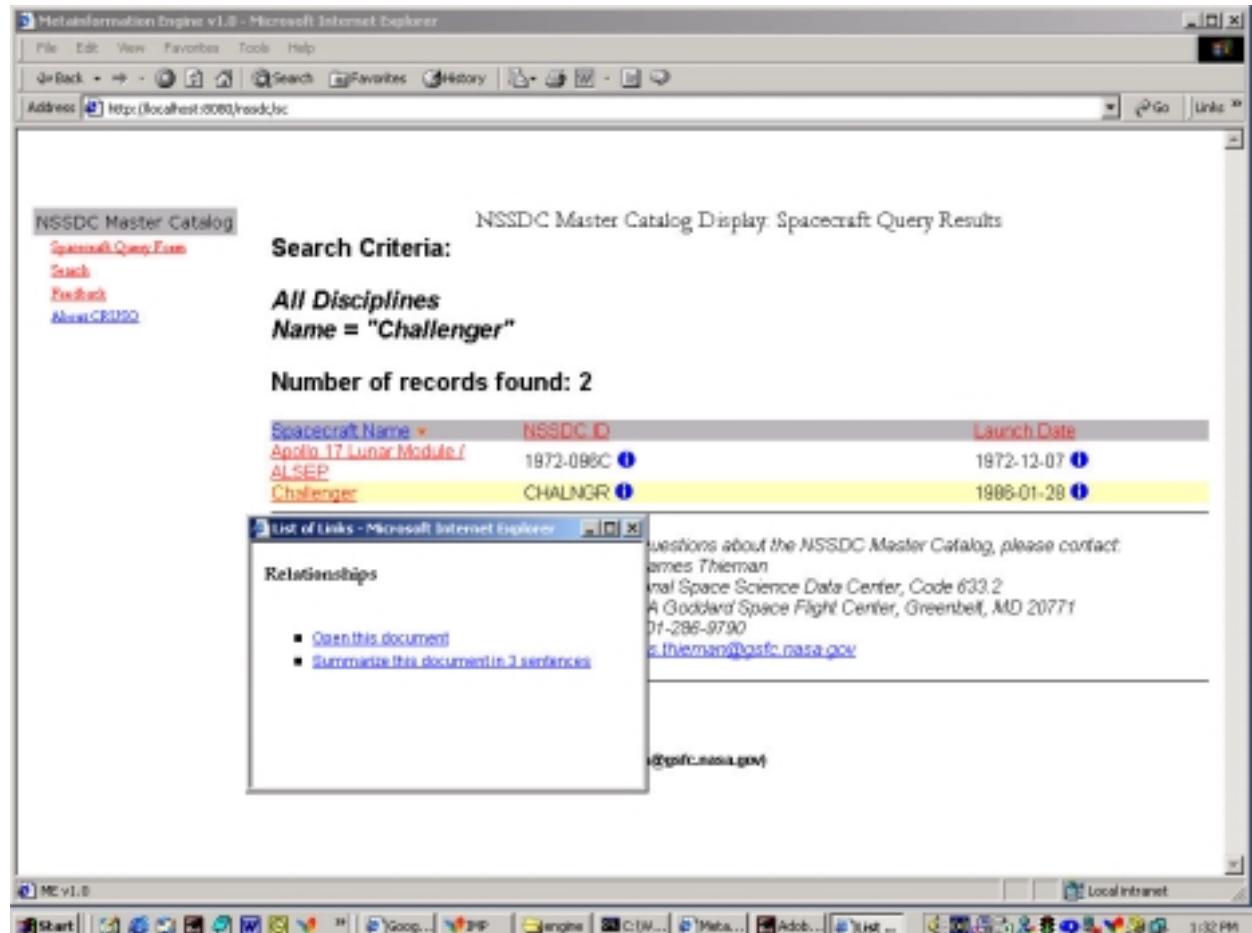


Figure 5: A screenshot of our current Digital Library Service Integration prototype. It integrates two independent digital library systems: NASA's National Space Science Data Center (NSSDC) Master Catalog and the AI Summarizer from the University of Arizona. The Metainformation Engine (ME) parsed the NSSDC document, adding its own link anchors (indicated by the circled "1" in the 2nd and 3rd columns). The second column contains NSSDC document identifiers. The third column contains launch dates. The original NSSDC system marks neither as a link anchor. When the user clicks on the document identifier "CHALNGR," the ME generates a list of two links for document identifiers (for elements of type "document"). The first will prompt the NSSDC system to display this document. The second will prompt the ME to pass the CHALNGR document to the external AI Summarizer system. Clicking on (an element of type) launch date generates a separate list of links to services relevant to that kind of element.

Figure 5 presents a screen shot from a current prototype of the ME for the Digital Library Service Integration project. This is the first prototype building towards the mockup illustration in Figure 2. The current prototype integrates two digital library systems: NASA's National Space Science Data Center (NSSDC) Master Catalog [<http://nssdc.gsfc.nasa.gov/>] and a document summarizer system, AI Summarizer, developed at the University of Arizona [Chau et al. 2002]. Users make queries into the space science data base from a query form. The NSSDC wrapper parses the query result, identifying NSSDC documents and launch date elements. The ME added

supplemental anchors on the document for these elements. The user clicked on one of these anchors. The ME then inferred the list of links shown from its base of relationship rules for the kind of element selected.

We note that the user interface is a prototype only. As part of our development we shall experiment with several different ways of presenting link anchors, sets of links and link descriptions.

5.4.2 Incorporating Just-in-Time Virtual Document Support

In order to provide WYWWYWI metainformation to virtual documents, we are extending the ME with several additional components or enginelets to support regeneration, re-location and re-identification (see section 4.1). We describe two of these components here: The Virtual Document Manager and the Regeneration Engine. [Zhang 2004] describes the full extended architecture.

Dynamically-generated virtual documents differ from static documents in many aspects. For example, a virtual document only requires specifications and other related document information (such as the document identifier, creation template and the commands invoking a system to regenerate them). A static document requires that a system stores the entire content. Once a static document is generated, the generation date, time, file size and content are consistent. This information could also be used to indicate if a static document is modified or a new version is created. A virtual document does not have any persistent status [Watters 1999].

A challenging issue is developing persistent identifiers both for virtual documents and for virtual elements, which could appear in multiple virtual documents [Zhang 2004]. We have assigned this as a task for the wrappers. When parsing virtual documents, they must (re)assign virtual document and element identifiers. We are achieving this on an *ad hoc* basis for certain systems, and are researching ways to systematize this and make it a relatively easy task for developers.

The ME uses persistent unique identifiers (ID) with the following format:

Virtual document identifier = (system ID, command ID, parameter value)

Virtual element identifier = (virtual document ID, element name)

The Virtual Document Manager (VDM) generates a persistent virtual document identifier by maintaining a list of system-specific commands and parameters in the database. A skeletal version of these IDs is designed in advance as part of developing the wrapper. The actual command and parameter values will be filled in at execution time by the VDM. For example, a database system may support SQL queries, but each query follows a well-defined format (that can be instantiated in an infinite number of ways).

The location of an anchor is generated dynamically when an anchor is placed in the virtual document and it is unique inside the document. XPATH expressions are used to address the anchors inside the document, and XPOINTER expressions are used to express arbitrary selections.

The Regeneration Engine (RE) serves three important functions. First, the RE gets the necessary commands and parameters from the RE database according to the virtual document ID. Second, to regenerate documents it sends commands to the appropriate system wrapper for execution and gets back resulting virtual documents in XML format. Third, it compares the newly-generated virtual document with the history information stored in RE database to *revalidate*

it. The RE supplies some “sameness” criteria to determine whether the regenerated document is the same as one it has previously encountered. Following are some sameness criteria, listed from very rigid to very flexible:

- The file content and structure should be exactly the same (very rigid).
- The file’s structure remains the same, but the content could be different. For example, element values such as the current date or current stock price may differ from the last time the document was generated, but these elements will be in the same relative location within the document as before.
- Some critical sections of the document should not change; other sections may.
- As long as the query is the same one that generated it, the system treats it as the same document (very flexible).

Which criterion the ME uses depends on the system or user requirements. Generally speaking, the more rigid the criterion is, the more information needs to be stored by the RE.

Elements of interest and anchors for link-based services are re-located primarily through structural analysis during parsing by the wrapper and only secondarily through content-based (lexical) analysis. With structural analysis, when an element’s value changes, the ME still can relocate and re-identify the element, avoiding the “dangling link” problem [Davis, 1999]. (For link-based service anchors declared by users inside virtual documents, which cannot be found through either structural or regular lexical analysis, we deploy other lexical techniques, which Davis pioneered.)

5.4.3 Incorporating Content-based Analysis

The ME architecture performs content-based analysis in two phases, first to find additional link anchors for potential elements of interest, and second to add content-based relationships to the list of links the ME generates. Currently we only perform lexical analysis to analyze the content.

We use Wu’s Noun Phrase Extractor [Wu 2000] to find the (root forms of) noun phrases within textual content. Then we compare each phrase to a combined master thesaurus from all systems registered with the ME. The master thesaurus will contain the URL to all references for recognized key phrases in external systems. Any phrases found in the master thesaurus will be candidates for link anchors. When the user clicks on one, the ME provides one link to thesaurus entry within the list of links returned. (In order not to overwhelm the user with link anchor choices, the ME may filter these and display only a subset as link anchors, or even display none visibly as link anchors but allow the user to click on any as if they were highlighted as anchors [Lewis et al. 1996].)

The master thesaurus maintains the domain for each noun phrase when available (e.g., business, medical) to assist in further filtering when the user’s interests are known. The domain can also be displayed as part of the label within the list of links to help guide the user.

As we extend the ME for more sophisticated content-based analysis and move into languages other than English, we will need the services of analogous tools that others have developed. The ME can incorporate these as enginelets or call them as an external service.

5.4.4 Related Research

System developers and the general public are starting to see general lexical-based linking services (e.g., Microsoft's Smart Tags) so they are starting to become aware of the potential for plentiful, general linking. No one, however, is applying general linking at the class-level within systems to effect integration and to support user interaction, as our relationship rules enable.

A few commercial products have appeared which generate anchors and links, such as NBC-Interactive's QuickClick, which actually has been withdrawn. QuickClick found its links in two ways. First, it had a list of predefined keywords that it recognized and a set of relevant links for each, such as company names. Second, it found relevant links through a standard keyword search. This is much less powerful than the ME's approach to structural linking.

Digital Library Service Integration

In the digital library field, for example, systems integration through linking is quite different from content interoperability. Very little research appears to exist concerning digital library service integration.

The Open Archives Initiative develops and promotes interoperability standards that aim to facilitate the efficient dissemination of content [<http://www.openarchives.org>]. While OAI and its protocol enables interoperability of content, DLSI concentrates on interoperability of services and presenting all relevant services to a user for a particular element of interest. OAI does not provide context-sensitive linking services, which DLSI will provide.

SFX [Van de Sompel 1999a,b,c; Van de Sompel & Beit-Arie 2001a,b] is the only approach we have found that is similar to the ME. Both SFX and the ME can generate a set of context-sensitive links. SFX operates primarily within citation environments. When the user clicks on an SFX-button (anchor) inserted by a citation, SFX looks up destination collections that contain the cited resource from a set of registered collections. SFX uses the OpenURL protocol to record metadata parameters for citations [Van de Sompel et al. 2000]. While primarily used with traditional online library resources, SFX's approach and the OpenURL standard are being generalized to include links to other kinds of digital library services and collections and indeed other non digital library systems [Van de Sompel & Beit-Arie 2001b].

Collections and services can automatically insert SFX-buttons into citation lists. It seems that they primarily generate these anchors at the same time they generate query results from a search. In contrast, the ME's approach requires few or no changes to the collection or service. The collection or service passes pages to the ME for further processing. ME system wrappers identify elements to link and the ME automatically inserts anchors over these elements. (Nothing precludes systems from post-processing documents as the ME does to include SFX-buttons; we just have found no published evidence that any do this yet.)

We note that the ME could integrate SFX as a citation service, with an SFX wrapper passing information to and from SFX in OpenURL format.

Hypermedia Engines and Web Systems

The ME also differs from most hypermedia linking engines [Anderson 1997, Carr et al. 1998, Davis et al. 1992, Grønbæk & Trigg 1999], which rely on user-declared manual linking or keyword search.

A few hypermedia engines execute independently of a system with minimal modifications to it, and provide the system's users with link-based services. Few approaches provide transparent integration as the ME does. Notable projects include Microcosm's Universal Viewer [Davis et al. 1994], Freckles [Kacmar 1993, 1995], the Distributed Link Service [Carr, 1995] and the OO-Navigator [Garrido and Rossi 1996; Rossi et al. 1996].

Microcosm's Universal Viewer and Freckles seamlessly support an application's other functionality but provide only manual linking. The Distributed Link Service (DLS) collects sets links ("linkbases") which applications or users can choose to activate. This enables links to be activated for specific contexts or domains. Then whenever a Web browser displays a page, the DLS link service searches each of the activated linkbases and retrieves links relevant to the items on that page. DLS links are specified manually for specific objects within a specific document, but can also be specified for a particular text phrase that could appear in any document. (We note that DLS does not deal with the "dangling link" problem of link endpoints no longer being available when documents are modified.) OO-Navigator comes the closest to our approach, providing a seamless support for dynamic linking within computational systems that execute within a single Smalltalk environment [Garrido and Rossi 1996]. This approach meets our goal of supplementing Smalltalk systems without altering them. Our approach applies to any Web-based system.

Web Database Systems

We note that several Web sites automate linking for database application systems. For example, electronic shopping systems will fill a template such as a catalog page, adding specific links for whichever product appears there. Several digital library query engines similarly add links to query results returned from a database. The ME provides a generalized approach that will function beyond database systems. Also, anchors within Web systems tend to have a single link, while anchors within ME-enhanced systems typically produce a list of relevant links, each representing a different relationship on the anchor's element of interest.

Systems Integration

System Integration (SI) enables the cooperation of multiple software modules. It encompasses a host of activities that are aimed at accessing data and programming logic in an environment characterized by distributed heterogeneous systems. The goal of SI is to utilize various autonomous systems in concert so that they support the achievement of a common goal, by providing an integrated set of data and services [Barret 1996]. It is often difficult to successfully carry out systems integration with systems that were developed independently with no thought to future integration [Nilsson, 1990]. Lightweight or loosely coupled systems integration through linking contributes an approach to systems integration that should facilitate widespread WYWYWI through the ME. In what follows, we contrast the lightweight integration approach of the ME to some common middleware architectures for systems integration.

Many systems are not designed to provide open and easy access to their data or functionality. This is often the problem faced when systems to be integrated belong to different, autonomous organizations. Intra-organizational integration provides an opportunity to enforce some degree of standardization or compliance to the systems that must be integrated. This in turn allows the SI solution to rely on proprietary protocols and fixed application program interfaces (APIs). However, integration architectures for inter-organizational system are not able to impose such constraints and therefore must rely on a mechanism that allows systems to cooperate without the

need for compliance with a rigid set of standards or protocols. This is why loosely coupled or lightweight systems integration approaches in distributed, heterogeneous, and independent collections of systems like the World Wide Web are needed to provide ubiquitous information to users.

The technology that provides integration between systems is often referred to as *middleware*. Integration technology is the mechanism that allows communication and the transfer of data among systems. It also allows one system to initiate actions on another system. This is the “technical” aspect of system integration that focuses on the applications and protocols utilized to enable communication between systems [Nilsson et al. 1990]. The various types of middleware include transactional, message-oriented, procedural, and object-component middleware [Emmerich 2000].

Emmerich [2000] provides a detailed discussion of the various middleware architectures, which is summarized in the following paragraphs. Transactional middleware applies the concepts of transactions from databases to providing phased commits to implement distributed transactions. This approach, however, introduces an enormous amount of overhead that is not necessary for non-transactional operations. Message oriented middleware enables clients and servers to communicate using messages and message queues. The client sends a message, which includes the service parameters, to the server component by inserting it into a message queue. The server responds to the client request with a reply-message containing the result of the service execution. Message systems rely on a communication mode that is asynchronous, and this is not appropriate for real-time information on the World Wide Web. The ME, on the other hand, can handle synchronous interaction by relying on the HTTP request/response model. This supports quick provision of information that is a result of a service request.

Procedural middleware utilizes remote procedure calls (RPC). In RPC server, programs export procedures and parameter types. The client can then invoke those procedures across the network. This method supports synchronous invocation, however the information (calls and parameters) must be automatically marshalled and unmarshalled. Marshalling and unmarshalling must also be done on the response sent back to the client from the server. Another disadvantage of procedural middleware is that interface definition for the server may have to be hard-coded into the client. These disadvantages lead to a rigid architecture and reliance on common proprietary protocols. In contrast, the ME utilizes XML documents, which can be transferred between components as uninterpreted byte strings, thus eliminating the need for marshalling and unmarshalling. However, the ME approach still requires that the target systems interface description be hard-coded into the wrapper.

In the object and component architecture, services advertise themselves in the service registry. Clients query the registry for service details and interact with the service using those details. A client looks up the reference to the factory object in the directory service. The client then asks the factory object to create a service object instance, this is done and the object reference to the server is returned to the client. The client uses the services object and destroys or releases it when it is finished [Vinoski, 2003]. This approach does not provide support for relationships between information items or objects (i.e., elements of interest). Instead it is focused on matchmaking between client and servers based on the functional interfaces described in the service registry. In contrast, the ME approach is organized around object relationships and the services that can be provided for various types of elements. The ME’s equivalent of service description registry is the relationship rules repository and this specifies the services that are available for various element

types rather than the description of parameters to be sent and methods to be initialized. The latter aspect is provided by the wrapper that communicates with the system.

6. COMBATING COGNITIVE OVERLOAD

Cognitive theory reminds us of the overhead that comes from needing to choose among multiple links, especially for novices not familiar enough with a domain to decide among these easily [Wurman, 1989]. The number of potential links that the ME could generate for a particular element on a screen could vary from several to well over a hundred, resulting in the well-known problem of cognitive overload [Conklin 1987; Halasz 1988; Thüring et al. 1995]. Users suffer from an overabundance of irrelevant information and they often selectively disregard information [Davis & Olson, 1985, Ackoff, 1967]. With a large number of links, filtering and ordering them is critical for effective use.

Filtering and rank ordering poses several challenges. First, it should be customized to each user's needs. Second, it should dynamically re-organize as the users advance through the system. Third, for the same user, support for multiple needs must be possible. A user may have several different tasks (needs) and the links should be re-organized depending on the user's current task.

Many researchers are working on adaptive approaches [Brusilovsky 1996, 2001] to customizing the links and content of Web pages for the user's level of experience, taking advantage of user models to capture this information. Often these code the adaptation specifications manually and then generate pages in real time based on these adaptation specifications. In the future, this approach could be combined with the conditions parameter on relationship rules.

Other filtering approaches such as that used with TaskMapper [Carroll et al. 1987], allow users to prune the links themselves.

The ME currently incorporates *collaborative filtering* to filter information based on people's evaluations or behaviors. The ME collects clickstream data as users navigate among documents. The collaborative filtering then generates recommendations using the following algorithm [Kostan et al., 1997; Herlocker et al., 1999; Im & Hars, 2001].

1. Calculate degree of similarity ("similarity index") between the current user and other users.
2. Identify a group of people ("reference group") who appear to share common interests with the current user. Their evaluations (or clickstreams) will be used for generating recommendations for the current user.
3. Calculate estimated evaluations for items that the current user has not seen (or evaluated). An estimated evaluation predicts the current user's evaluation on an item.
4. Rank order the items according to the estimated evaluations and select the top n items to recommend.

Collaborative filtering, however, will only solve certain problems, and relies on many people selecting the same elements of interest within application systems and getting similar sets of link choices. In future research we shall explore additional approaches to user modelling and customization, including methods that directly involve the user. Users could provide additional information about their current tasks and preferences. Users also could tell the system which links they find useful.

A further cognitive issue is “disorientation” [Conklin 1987; Thüring et al. 1995]. In his commentary as referent for [Bhargava et al. 1988] at the 1988 International Conference on Information Systems, George Widmeyer of the University of Michigan, stated that the WYWYWWI Principle can lead to the HYGWYGYGT (how you got, where you got, when you got there) complex. Good navigation design [Christodoulou et al. 1998, Schwabe et al. 2001] and user interface design are necessary to keep the user oriented in a complex Web space with many link choices.

The design of the ME’s interface and interaction is equally important. As we continue to extend the ME and the way it presents metainformation and linking choices, we will need to continuously evaluate its usability.

7. RELATIONSHIP ANALYSIS

How will developers determine the non-obvious structural relationships and metadata to incorporate into relationship rules? This is the role of Relationship Analysis (RA).

RA is a systematic and rigorous elicitation technique to discover the relationship structure of the problem domain. . An analyst, designer or developer would use it during the early stages of the development process to discover the relationship structure of a new or existing system.

During the system analysis phase, components are determined through identifying the system’s entities and relationships. Informal guidelines exist to help identify entities or objects [Chen, 1976, Rumbaugh, 1991, Booch, 1994]. However, prior to RA, no guidelines existed to analyze a system domain in terms of its relationship structure. Determining a system’s relationship structure was an implicit process. No defined processes or diagrams exist to explicitly or systematically assist in eliciting relationships or documenting them in traditional Entity-Relationship (ER) Diagrams [Beraha & Su, 1999] or object-oriented class diagrams. In addition, while the Unified Markup Language (UML) provides the class diagram for the designer to document objects and relationships *once identified*, it includes no diagram that assists the analyst in identifying the relationships and then communicating them to the designer. Furthermore, relationships are a key component lightly addressed by ER and class diagrams. These diagrams capture a limited subset of relationships and leave much of the relationship structure out of the design and system model. While models generally are meant to be a limited representation of a system, this incomplete relationship specification is not by design, but rather owing to a lack of any methodology to determine them explicitly [Bieber & Yoo, 1999, Bieber, 1998]. As a result, many analyses miss aspects of the systems they represent.

Relationship Analysis addresses these concerns and provides a rigorous and systematic technique to identify the relationship structure of a problem domain and helps fill a void in the systems analysis process. RA provides both a UML-style template and diagram [Catanio, 2004]. We are designing RA as a standard extension to current systems analysis methodologies, and so it seamlessly fits within the UP and in UML modeling. RA fills this void by providing a technique to explicitly identify the relationship structure of the problem domain [Catanio, 2004].

Relationships can be determined for anything in the system that some stakeholder may be interested in. Elements of interest form the endpoints for the relationships and relationships form the links. This type of information provides the infrastructure needed for link-based services. The infrastructure is based in a thorough taxonomy of the relationships found in a computer system’s domain [Yoo 2000, Yoo & Bieber 2000a,b]. Each of the taxonomy’s sixteen categories has a

series of exploratory questions to reveal or “elicit” the implicit set of relationships. Section 7.1 presents the highest-level RA taxonomy. Section 7.2 describes our elicitation procedure.

. 7.1 RA’s Generic Relationship Taxonomy

Table 1 presents RA’s generic, domain-independent relationship taxonomy.

Generalization/Specialization	
<i>Self</i>	Characteristic Descriptive Occurrence
<i>Whole-part /Composition</i>	Configuration/Aggregation Membership/Grouping
Classification/Instantiation	
<i>Comparison</i>	Equivalence Similar/Dissimilar
<i>Association /Dependency</i>	Ordering Activity Influence Intentional Socio-organizational Temporal Spatial

Table 1: RA’s 16 Generic Relationships

These relationship categories were developed based on a very extensive literature review [Yoo 2000] and strenuous trial-and-adjustment prototyping. [Yoo 2000] compares RA’s taxonomy with 10 other domain-specific taxonomies in detail, with additional comparisons with over 20 others. RA’s categories encompass all of these other taxonomies’ relationships. This includes, for example, object-oriented analysis [Martin and Odell, 1995] (which provides RA’s generalization/specialization, whole-part, classification/instantiation and association relationship classifications).

Generalization/specialization relationships concern the relationships between objects in a taxonomy [Borgida et al., 1984, Brachman, 1983, Smith and Smith, 1977]. Self relationships include characteristic, descriptive, and occurrence relationships. They are especially useful in identifying non-obvious metadata.

Whole-part/composition relationships include configuration/aggregation relationships based on configuration aspect of the whole-part relationships, and membership/grouping relationships [Brodie, 1981, Motschnig-Pitrik and Storey, 1995] based on membership aspect of the whole-part relationships [Henderson-Sellers, 1997, Odell, 1994]. Classification relationships connect an element of interest and its class or its instance.

Comparison relationships break down into similar/dissimilar and equivalence relationships, involving such relationships as in thesaurus or information retrieval [Belkin and Croft, 1987, Neelameghan and Maitra, 1978]. Association/dependency relationships break down into ordering, activity, influence, intentional, socio-organizational, spatial and temporal relationships. The term association and dependency could be used interchangeably, because every association involves

some concept of dependency [Henderson-Sellers, 1998]. Because association is defined as a relationship that is defined by users, there could be no fixed taxonomy for it. The association relationship taxonomy is fluid compared with other relationships. The current association relationship taxonomy is based on our observations, analyses, ontologies [Mylopoulos, 1998], and existing classifications [Henderson-Sellers, 1998].

Ordering relationships involve some kind of sequence among items. Activity relationships are created by combining SADT activity diagrams [Mylopoulos, 1998] and case relationships [Fillmore, 1968] to deal with relationships associated with activities or actions abstractly. This relationship could cover any activities that involve input or output, and deal with agents and objects involved in the activities. Influence relationships exist when one item has some power over the other items. Intentional and socio-organizational relationships could be identified in intentional and social ontologies respectively. Temporal [Allen, 1983, Frank, 1998] and spatial [Cobb and Petry, 1998, Egenhofer and Herring, 1990, Rodriguez et al., 1999] relationships deal with temporal and spatial perspectives, respectively.

Each relationship category can be further broken down into lower levels of detail, from which we derived a basic set of brainstorming questions. [Yoo 2000] details each lower-level category and the literature from which we derived each.

7.2 Conducting a Relationship Analysis

We begin by identifying the elements of interest for which we want to provide structural relationships and metadata. For existing systems, we can look at screen shots to identify the elements of interest that a user might want to request metainformation about. When designing a new system, we can identify elements of interest (entities) from the use cases. For each element of interest, an analyst asks a series of questions to elicit characteristics about it and the relationships around it, which actually often leads to discovering additional elements of interest these connect.

Table 2 gives a sample of these exploratory questions that an analyst uses to elicit domain information from the domain expert. Each set of questions is derived from the lower levels of detail for each relationship in the taxonomy, described in [Yoo 2000]. For brevity, the subset of questions in Table 2 are rather condensed and highly generic. The analyst would tailor them to each item of interest for the particular domain. For example, the descriptive relationship prompts analysts to ask whether an item of interest has “a definition, explanation, set of instructions or illustrations available within or external to the system.” (These are all lower-level categories for the generic relationship “descriptive.”) The analyst clearly should ask each of the questions individually, and in a way that makes the most sense to the particular domain expert.

Generalization/ Specialization	Is there a broader term for this item of interest? Is there a narrower term for this item of interest?
Descriptive	Does an item of interest have a description, definition, explanation, or a set of instructions or illustrations available within or external to the system?
Configuration/ Aggregation	Which components consist of this item? What materials are used to make this item? What is it a part of? What phases are in this whole activity?
Similar/ Dissimilar	Which other items are similar to this item of interest? Which others are opposite to it? What serves the same purposes as this item of interest?
Activity	What are this item's inputs and outputs? What resources and mechanisms are required to execute this item?
Influence	What items (e.g., people) cause this item to be created, changed, or deleted? What items have control over this item?
Intentional	Which goals, issues, arguments involve this item of interest? What are the positions and statements on it? What are the comments and opinions on this item? What is the rationale for this decision?
Socio- organizational	What kinds of alliances are formed associated with this item of interest? Who is committed to it in the organizational structure? Who communicates with it or about it, under what authority and in which role?

Table 2. Sample exploratory questions emanating from RA's generic relationships in Table 1. See [Yoo 2000; Yoo & Bieber 2000a,b] for additional questions, including those for the other relationships.

Figure 6 illustrates a selection of the relationships discovered for the element “book”. A fuller analysis can be found in [Yoo & Bieber 2000b].

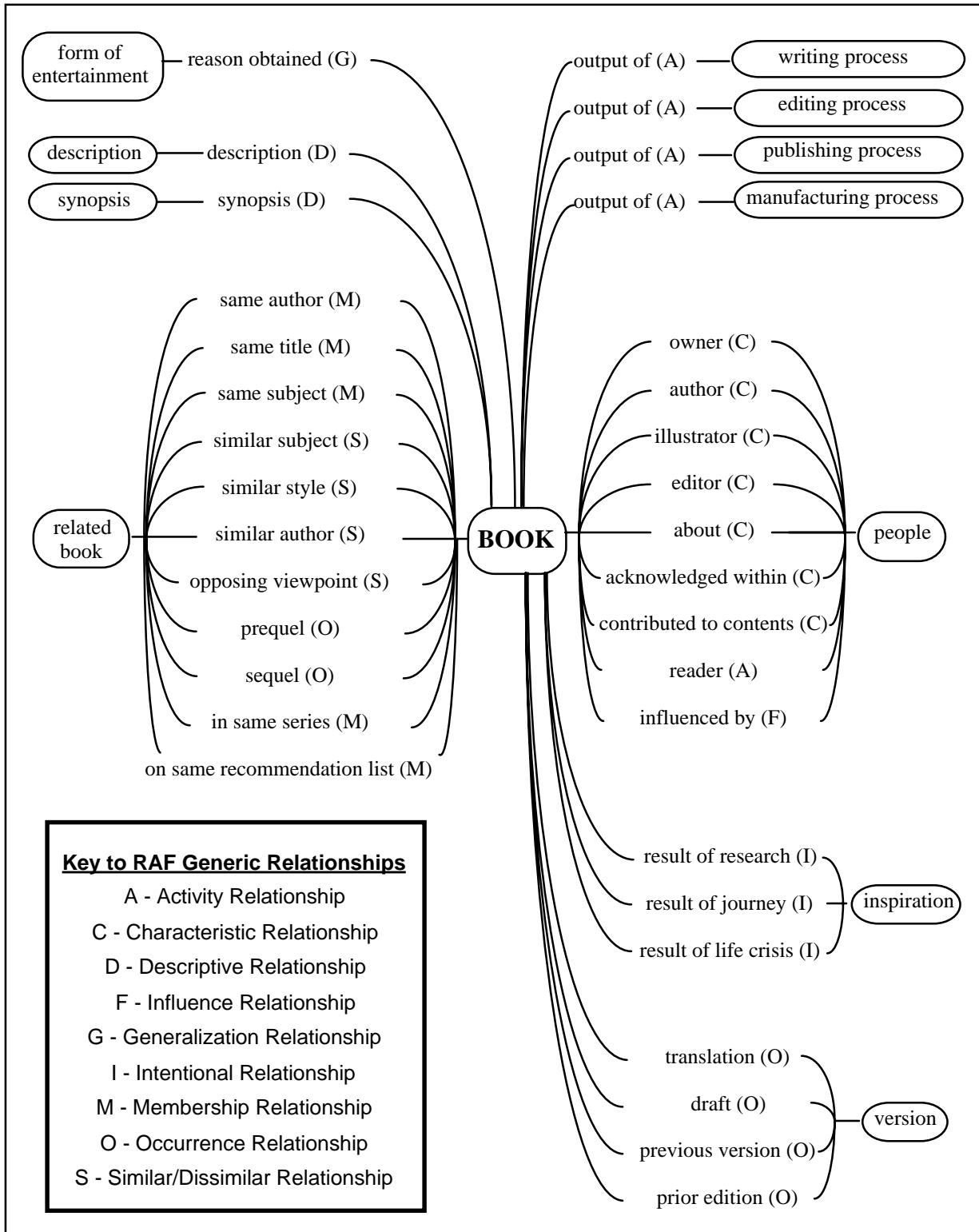


Figure 6. A subset of the relationships around books found through a Relationship Analysis (RA). Each is annotated with the relationship category associated with the RA questions used to discover them from Table 1's relationship taxonomy.

Thus, analysts collecting information about the functionality of a system using use-cases could also use RA to gather relationship-specific information. These two approaches should be used iteratively to gather the wealth of information needed to build and deploy successful systems. Just as use-cases identify the primary elements (entities) to get RA started, the relationships elicited and their endpoint elements can effectively identify new use-cases. Together, these two approaches when used during the critical requirements analysis phase can provide a depth of information never before available to analysts, designers and developers.

RA provides an important, missing tool for analysts and designers to embrace the WYWWYWI principle. RA comprehensively identifies metainformation for system designs, and for integrating existing systems into the ME infrastructure. As metainformation support moves from Web systems to supplementing real world objects (e.g., through augmented reality), our future research should show RA to be robust enough to reveal the inter-relationships within these non-computerized domains as well.

8. UBIQUITOUS WYWWYWI

As developers buy into the WYWWYWI Principle and wish to start vastly increasing the number of links potentially available, the ME approach could gather momentum and gain universal acceptance. We envision the following activities to occur to support this movement, and in fact, this outlines much of our future research plans.

ME as a Web Service: We plan to offer the ME as a general metainformation service. When systems have a page of information to display, they can send the page to the ME, which will return it supplemented with link anchors and links to metainformation. Systems will be able to register wrappers, relationship rules, thesauri and glossaries, and link-based services, which the ME could then use and give access to.

Wrappers: We envision the general development of wrappers for the everyday systems used within the home, businesses, education and other organizations. Wrappers will parse documents and screens to identify elements of interest (even if only through content-based analysis). The growth of XML markup will greatly assist this effort.

Wrappers will be developed for e-commerce applications, legacy systems, enterprise resource planning systems (ERP), architectural systems, scientific systems, computer conferencing systems, educational support systems, library systems, government service systems, and desktop productivity software products, among others. The general public will use many of these systems; others will be used in organizations behind a firewall. Many wrappers will be implemented as plug-ins, compatible with everyday browsers.

Some developers will be motivated by profit and may charge for these wrappers or bundle them for free with their software systems. Others may not charge for them but embed their own links or even advertisements. We also would hope that an open-source movement would evolve that provides such wrappers to the general public for free.

Relationship Rules: We envision ubiquitous access to relationship rules. Developers will come to see relationship rules both as free publicity for their systems (bringing users from the outside to them), as well as free functionality expanding their systems (giving users additional services for free, making their systems more satisfying). Relationship rules also provide an easy way to streamline systems. Many of the links could take users directly to other screens and functions

within large systems, thus giving the user more direct control over system navigation and direct access to exactly what they want, when they want it.

Lastly, relationship rules implement lightweight systems integration through linking. In many cases this can provide a relatively inexpensive solution to systems integration for many organizations.

Central registries of relationship rules will be developed that any ME can access. Relationship rule repositories may need to remain distributed, the registries will need to be centralized to a practical degree.

Relationship Analysis: We envision relationship analysis being widely used by systems analysts and developers to determine which structural relationships to include as relationship rules. We are developing the first generation of tools to assist in Relationship Analysis, making it easy to apply for both novices and expert analysts.

Thesauri and Glossaries: we envision the growth of thesauri and glossaries, and registries that ME implementations can access. The thesauri and glossaries will remain distributed, but the registries will need to be centralized to a practical degree.

Content-based Analysis: Researchers and commercial developers will continue to produce sophisticated lexical analysis programs that work with the glossaries to produce lexical relationships, as well as practical techniques for identifying elements within non-textual content.

Link-based Services: We envision a blossoming of link-based services, many implemented through browser plug-ins. These will include easy-to-use authoring tools. It will become commonplace for people both at home and in the workplace, to build and share annotations, guided tours, bookmark lists, and personalized overviews of information.

9. IN CLOSING

Science fiction writers fantasize about people who vividly can see the relationships among people and objects. Designers should do this naturally for their information domains. But they rarely do. Few designers explicitly think about their systems' interrelationships and whether users should access and navigate them directly. Why not? In part, it has not occurred to many designers and developers to incorporate a plethora of metainformation. Most designers and developers do not have a WYWWYWI mindset; they and their users have seen few examples and do not demand this comprehensiveness yet. In part, people do not have the time to reengineer existing systems, whether they be legacy systems without Web interfaces or Web systems with limited link choices, given the never-ending queues of other projects waiting to be implemented. Developers furthermore have few tools and techniques for designing and incorporating metainformation and link-based services easily. Developers will not do this until it is natural to conceive and easy to implement.

The combination of Relationship Analysis, the Metainformation Engine and ubiquitous tools for link-based services could revolutionize how people—both developers and users—conceive of information access. We look forward to the day when we can point to any object, on or off the screen and say "This looks interesting. Tell me more about it. What is it? How can I use it? What do I need to know to use it? Can I modify it? What can I do once I have deployed it? How does it differ from other similar objects?" These are all relationships and help us understand the full

context of things, in which we are interested. We all should be able to access whatever information we want, when we want it.

ACKNOWLEDGEMENTS

We are honored to acknowledge Steven O. Kimbrough of the Wharton School of Business at the University of Pennsylvania, who originated the term WYWYWI in 1988. This vision has guided our research for the last 15 years. We gratefully appreciate partial funding support for this research by the United Parcel Service, the New Jersey Center for Pervasive Information Technology, the New Jersey Commission on Science and Technology, and the National Science Foundation under grants IIS-0135531 and DUE-0226075.

ABOUT THE AUTHORS

Michael Bieber is an Associate Professor of Information Systems. He teaches both on-campus and in the distance learning program, often combining the students in both modes. Dr. Bieber is conducting research in several related areas: participatory collaborative learning, asynchronous learning networks, distance education, infrastructures for educational software, lightweight systems integration, digital library integration, relationship analysis (as part of the software engineering process), Web engineering, link-based services, automatic link and metainformation generation, hypermedia, community informatics and virtual communities. He co-directs NJIT's Collaborative Hypermedia Research Laboratory. He holds a Ph.D. in Decision Sciences from the University of Pennsylvania.

Joseph Catanio is a Ph.D. student in the Information Systems Department at NJIT. He has worked as a software engineer for 15 years doing embedded systems, application, and Web development. As a researcher, Joe has developed a process to explicitly identify and document the relationship structure of a problem domain. This process, Relationship Analysis is being positioned as part of the software engineering process. He holds a BS in Electrical Engineering from Rutgers University, an MS in Computer Science from NJIT, and is on target to earn his Ph.D. in Information Systems from NJIT in May 2004.

Roberto Galnares is currently the Chief Technology Officer of a company in the insurance sector. Dr. Galnares has conducted research on dynamic metainformation generation, meta-languages, web engineering, distributed applications, and object-oriented vocabularies. His dissertation was based on the design and implementation of a Dynamic Hypermedia Engine, an earlier version of the MetaInformation Engine. He performed part of his academic training as researcher at the IBM Watson Research Center in New York. He holds a Ph.D. in Computer and Information Science from the New Jersey Institute of Technology.

Nkechi Nnadi is a Ph.D. student in the Information Systems Department at NJIT. Her research interests include component-based software architectures, semantic integration of systems, and resolving semantic differences in ontologies.

Li Zhang is a Ph.D. student in the Computer Science Department at NJIT. Her research interests include dynamic generation of virtual documents and automatically providing hypermedia support to these.

REFERENCES

- Ackoff, R.L. (1967), Management Misinformation Systems, *Management Science*, 14, (4), 147-156.
- Allen, J (1983). Maintaining Knowledge about Temporal Intervals, *Communication of the ACM*, 26(11), 832-843.
- Anderson, K. (1997). Integrating Open Hypermedia Systems with the World Wide Web. *Hypertext'97 Proceedings*, ACM Press, New York, NY, 157-166.
- Barrett, D.J., et al. (1996). A Framework for Event-Based Software Integration, *ACM Transactions on Software Engineering and Methodology*, 5(4).
- Belkin, N. and Croft, W. (1987) Retrieval Techniques, *Annual Review of Information Science and Technology* (ARIST), Vol. 22, 1987, Chapter 4, 109-131.
- Beraha, S., and Su, J. (1999), Support for Modeling Relationships in Object-Oriented Databases, *Data & Knowledge Engineering*, Vol. 29, No. 3, 227-257.
- Bhargava, Hemant K., Michael P. Bieber and Steven O. Kimbrough (1988). Oona, Max, and the WYWYWI Principle: Generalized Hypertext and Model Management in a Symbolic Programming Environment, in *Proceedings of the Ninth International Conference on Information Systems*, Minneapolis, 1988, pages 179-192. [on-line]
<http://web.njit.edu/~bieber/pub/bbk88.pdf>
- Bieber, M. (1998), Hypertext and Web Engineering, *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia*, ACM Press, 277-278.
- Bieber, Michael and Steven O. Kimbrough (1992), On Generalizing the Concept of Hypertext, *Management Information Systems Quarterly*, 16(1), 77-93.
- Bieber, Michael and Steven O. Kimbrough (1994), On the Logic of Generalized Hypertext, *Decision Support Systems* 11, North Holland, 241-257.
- Bieber, Michael and Fabio Vitali (1997). Toward Support for Hypermedia on the World Wide Web *IEEE Computer* 30(1).
- Bieber, Michael, Fabio Vitali, Helen Ashman, V. Balasubramanian, and Harri Oinas-Kukkonen (1997). Fourth Generation Hypermedia: Some Missing Links for the World Wide Web *International Journal of Human-Computer Studies* 47, 1997, 31-65.
- Bieber, Michael and Joonhee Yoo (1999). Hypermedia: A Design Philosophy, *ACM Computing Surveys*, 31(4es), 1999.
- Booch, G. (1994), *Object-Oriented Analysis and Design*, Second Edition, Benjamin/Cummings Publishing Company, California.
- Borgida, A., Mylopoulos, J., and Wong, H. (1984) Generalization/Specialization as a Basis for Software Specification, *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Ed. Brodie, M., Mylopoulos, J., and Schmidt, J., 87-117.
- Brachman, R. (1983). What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks, *IEEE Computer*, 30-36.

- Brodie, M. (1981). Association: A Database Abstraction for Semantic Modelling. Entity-Relationship Approach to Information Modeling and Analysis, P.P. Chen (ed.), ER Institute 583-608.
- Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. In P. Brusilovsky and J. Vassileva (eds.), User Modeling and User-Adapted Interaction 6(2-3), 87-129.
- Brusilovsky, P. (2001). Adaptive Hypermedia. User Modeling and User Adapted Interaction 11(1/2), 87-110.
- Burton, John K., D. Michael Moore and Glen A. Holmes (1995). Hypermedia Concepts and Research: An Overview. Computers in Human Behavior 11: 345-369.
- Carr, L. A.; De Roure, D., Hall, W., Hiil, G. (1995). The Distributed Link Service: A Tool for Publishers, Authors and Readers. Proceedings of the 4th International World Wide Web Conference, Boston, Massachusetts, December 11-14, 1995.
- Carr, L. A., Hall, W., Hitchcock, S. (1998). Link Services or Link Agents?, Proceedings of ACM Hypertext '98, Pittsburgh PA, 113-122.
- Carroll, J.M., Herder, R.E., & Sawtelle, D.W. (1987). TaskMapper and the Advisory Interface Dilemma. In H.-J. Bullinger & B. Shackel (eds.), *Proceedings of Second IFIP Conference on Human-Computer Interaction Interact87* (Stuttgart, September 1-4). Amsterdam: North-Holland, 973-978.
- Catanio, Joseph (2004), Team-Based Relationship Analysis, Ph.D. Dissertation, New Jersey Institute of Technology.
- Chau, Michael, H. Chen, Jialun Qin, Yilu Zhou, Yi Qin, Wai-Ki Sung, Daniel McDonald (2002), Comparison of Two Approaches to Building a Vertical Search Tool: A Case Study in the Nanotechnology Domain, Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'02), Portland, Oregon, July 14-18, 2002.
- Chen, P. (1976), The Entity-Relationship Model – Toward a Unified View of Data, ACM Transactions on Database Systems, Vol. 1, No. 1.
- Christodoulou, S., G. Styliaras and T. Papatheodourou (1998). Evaluation of Hypermedia Application Development and Management Systems, Proceedings of ACM Hypertext '98 Conference, Pittsburgh, 1-10.
- Cobb, M. and Petry, F. (1998). Modeling Spatial Relationships within a Fuzzy Framework, Journal of the American Society for Information Science, 49(3), 253-266.
- Cockburn, A. and Jones, S. (1996). Which Way Now? Analysing and easing inadequacies in WWW navigation, Int. J. Human-computer Studies, 45: 105-129
- Conklin, J. (1987). Hypertext: An Introduction and Survey. IEEE Computer 20(9): 17-41.
- Cotkin, George (1996). 'Hyping the Text': Hypertext, Postmodernism and the Historian. American Studies 37: 103-116.
- Davis, G. B., and Olson, M.H. (1985). Management Information Systems: Conceptual Foundations, Structure and Development, McGraw-Hill Inc.
- Davis, H. C. (1995). Data integrity problems in an open hypermedia link service. Ph.D. thesis, Southampton University.
- Davis, H. C. (1999). Hypertext Link Integrity. ACM Computing Surveys 31(4).

- Davis, H., W. Hall, I. Heath, G. Hill and R. Wilkins (1992). Microcosm: an Open Hypermedia Environment for Information Integration. University of Southampton CSTR 92-15.
- Davis, H. C., S. Knight and W. Hall, (1994). Light Hypermedia Link Services: A Study of Third Party Application Integration.
- Dix, A. (1995). Dynamic Pointers and Threads. *Collaborative Computing*, 1(3): pp. 191-216.
- Egenhofer, M. and Herring, J. (1990). Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases, Technical Report, Department of Surveying Engineering, University of Maine.
- Emmerich, W. (2000), Software Engineering and Middleware: A Roadmap, May 2000, Proceedings of the Conference on the Future of Software Engineering.
- Fillmore, C.J. (1968). The case for cases in Universals in Linguistic Theory.
- Frank, A. (1998). Different Types of Times in GIS, Spatial and Temporal Reasoning in Geographic Information Systems, 1998, Eds. Egenhofer, M. and Golledge, R. Chapter 3, 41-62.
- Galnares, R. (2001). Augmenting Applications with Hypermedia Functionality and Metainformation. Ph.D. thesis, New Jersey Institute of Technology, Newark, NJ 07102.
- Garrido, A., and Rossi, G., (1996). A Framework for Extending Object-Oriented Applications with Hypermedia Functionality. *The New Review of Hypermedia and Multimedia*, 2: 25-41.
- Grønbæk, Kaj, Jannie Kristensen, Peter Ørbæk and Mette Eriksen (2003). ‘Physical Hypermedia’: Organizing Collections of Mixed Physical and Digital Material. *Hypertext 2003 Proceedings*, Nottingham UK, August 2003, 10-19.
- Grønbæk, K. and R. Trigg (1999). From Web to Workplace: Designing Open Hypermedia Systems, MIT Press.
- Halasz, F. (1988). Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communication of the ACM* 31(7): 836-855.
- Henderson-Sellers, B. (1997). OPEN Relationships-Compositions and Containments, *Journal of Object-Oriented Programming*, November/December 1997, 51-72.
- Henderson-Sellers, B. (1998). OPEN Relationships-Associations, Mappings, Dependencies, and Uses, *Journal of Object-Oriented Programming*, February 1998, 49-57.
- Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering, *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, ACM Press, New York, NY.
- Im, Il and Hars, Alexander (2001), Finding information just for you: Knowledge reuse using collaborative filtering systems, *Proceedings of International Conference on Information Systems (ICIS)*, New Orleans, Louisiana.
- Kacmar, C. (1993). Supporting hypermedia services in the user interface, *Hypermedia* 5(2), 85-101.
- Kacmar, C. (1995). A process approach for providing hypermedia services to existing, non-hypermedia applications, *Electronic Publishing: Organization, Dissemination, and Distribution* 8(1), 31-48.

- Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Good, N., and Riedl, J.(1997). GroupLens: Applying collaborative filtering to Usenet news, Communications of the ACM, 40(3), 77-87.
- Paul H. Lewis, Hugh C. Davis, Steve R. Griffiths, Wendy Hall, Rob J. Wilkins (1996). Media-based Navigation with Generic Links. Hypertext'96 Proceedings.
- Martin, J. and Odell, J. (1995) Object-Oriented Methods: A Foundation, Prentice Hall, Englewood Cliffs, New Jersey.
- Miller, J. Hillis. (1995). The Ethics of Hypertext. Diacritics 25.3: 27-39.
- Motschnig-Pitrik, R. and Storey, V. (1995). Modelling of set membership: The notion and the issues, Data & Knowledge Engineering 16, 147-185.
- Mylopoulos, J. (1998). Information Modeling in the Time of the Revolution, Information
- Neelameghan, A. and Maitra, R. (1978). Non-hierarchical associative relationships among concepts: Identification and Typology, Part A of FID/CR report no. 18, Bangalore: FID/CR Secretariat Document Research and Training Center.
- Nielsen, J. (1990). The art of navigating through hypertext. Communications of the ACM, vol. 33, no. 3, 196-310.
- Nielsen, Jakob. (1995). Multimedia and Hypertext: The Internet and Beyond. Boston: AP Professional.
- Nilsson, E.G.; Nordhagen, E.K.; Oftedal, G. (1990). Aspects of Systems Integration, Proceedings of the First International Conference on Systems Integration, 23-26 April 1990.
- Odell, J. (1994). Six different kinds of composition, Journal of Object-Oriented Programming, January 1994, 10-15.
- Oinas-Kukkonen, Harri (1998). What is a link? Communications of the ACM. 7; 41(7): 98.
- Rodriguez, M., Egenhofer, M. and Rugg, R. (1999). Assessing Semantic Similarities Among Geospatial Feature Class Definitions, Interop '99, Zurich, Switzerland, in: A. Vckovski (editor), Lecture Notes in Computer Science, New York.
- Rossi, G., A. Garrido and S. Carvalho (1996). Design Patterns for Object-Oriented Hypermedia Applications. Book chapter in: Pattern Languages of Programs II. J. Vlissides, J. Coplien and N. Kerth, eds. Addison- Wesley, 177-191.
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991) Object-oriented modeling and design. Prentice Hall, Englewood Cliffs, New Jersey.
- Schwabe, D., Rossi, G., Esmeraldo, L. and F. Lyardet. (2001). Engineering Web Applications for Reuse, IEEE Multimedia, Spring 2001, 2-12.
- Shneiderman, Ben, (2000). Universal Usability, Communications of the ACM 43(5), 84-91.
- Smith, J. and Smith, D. (1977). Database Abstractions: Aggregation and Generalization, ACM Transactions on Database Systems, 2(2), 105- 133.
- Thüring, Manfred, Jörg Hannemann and Jörg Haake (1995). Hypermedia and Cognition: Designing for Comprehension, Communications of the ACM 38(8), 57-69.
- Van de Sompel, H. and O. Beit-Arie. (2001). Generalizing the OpenURL Framework beyond References to Scholarly Works: the Bison-Fute Model. D-Lib Magazine, March 2001.

- Van de Sompel, H., and O. Beit-Arie (2001). Open Linking in the Scholarly Information Environment Using the OpenURL Framework. *D-Lib Magazine.*, August 2001.
- Van de Sompel, Herbert and Patrick Hochstenbach (1999a). Reference Linking in a Hybrid Library Environment, Part 1: Frameworks for Linking, *D-lib Magazine* 5(4).
- Van de Sompel, Herbert and Patrick Hochstenbach (1999b). Reference Linking in a Hybrid Library Environment, Part 2: SFX, a Generic Linking Solution, *D-lib Magazine* 5(4).
- Van de Sompel, Herbert and Patrick Hochstenbach (1999c). Reference Linking in a Hybrid Library Environment, Part 3: Generalizing the SFX solution in the SFX@Ghent & SFX@LANL experiment, *D-lib Magazine* 5(10).
- Van de Sompel, Herbert; Hochstenbach, Patrick and Beit-Arie, Oren (2000). OpenURL syntax description. Technical Report. [on-line: <http://www.sfxit.com/openurl/openurl.html>]
- Vinoski, S. (2002). Middleware “Dark Matter”, *IEEE Internet Computing*, September-October 2002.
- Vlahakis, V., J. Karigiannis, M. Tsotros, M. Gounaris, L. Almeida, D. Stricker, T. Gleue, I. Christou, R. Carlucci, N. Ioannidis (2001). ARCHEOGUIDE: First results of an Augmented Reality, Mobile Computing System in Cultural Heritage Sites, Virtual Reality, Archaeology, and Cultural Heritage International Symposium (VAST01), Glyfada, Nr Athens, Greece, 28-30 November 2001.
- Watters. C. and Shepherd, M. (1999). Research Issues for Virtual Documents. Workshop on Virtual Documents, Hypertext Functionality and the Web at the 8th International World Wide Web Conference.
- Wu, Yi-Fang (2000). Automatic Concept Organization: Organizing Concepts from Text Through Probability of Co-occurrence Analysis. Proceedings of the 11th ASIST SIG/CR Classification Research Workshop, Chicago.
- Wurman, R. S. (1989). *Information Anxiety*. Doubleday, New York.
- Yoo, Joonhee (2000). Relationship Analysis, Ph.D. Dissertation, Rutgers University, 2000.
- Yoo, Joonhee and Michael Bieber, (2000a). Towards a Relationship Navigation Analysis, Proceedings of the 33rd Hawaii International Conference on System Sciences, IEEE Press, Washington, D.C., January 2000.
- Yoo, Joonhee and Michael Bieber, (2000b), A Relationship-based Analysis, *Hypertext 2000* Proceedings, San Antonio, ACM Press, June 2000.
- Zhang, Li. (2004) Just-in-Time Hypermedia. Ph.D. Thesis, New Jersey Institute of Technology, Computer Science Department.