Adlet: An Active Document Abstraction for Multimedia Information Fusion

Shi-Kuo Chang and Taieb Znati

Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA

ABSTRACT

A new approach is described for the fusion of multimedia information based on the concept of **active documents advertising** *on the Internet, whereby the metadata of a document travels in the network to seek out documents of interest to the parent document, and at the same time advertises its parent document to other interested documents. This abstraction of metadata is called an* **adlet**, *which is the core of our approach. Two important features make this approach applicable to multimedia information fusion, information retrieval, data mining, geographic information systems and medical information systems: a) any document, including a web page, database record, video file, audio file, image and even paper documents, can be enhanced by an adlet and become an active document; b) any node in a nonactive network can be enhanced by adlet-savvy software and the adlet-enhanced node can co-exist with other non-enhanced nodes. An experimental prototype provides a testbed for feasibility studies in a hybrid active network environment.*

Keywords: Active document, information fusion, active index system

1 Introduction

The Internet has become an indispensable means of communication and collaboration among people at different levels and in various capacities. It provides access to a bewilderingly large number and variety of resources, including text, audio and video files, scientific data, retail products, network services, and

transcripts of conversations. Because of the scale and decentralized nature of this environment, the Internet has evolved into a chaotic repository of all types of information, making it difficult to locate resources of interest. If it is to continue to grow and thrive as new means of communication, new tools are needed to organize and support the Internet's resource discovery in a fashion that keeps pace with its exponential growth in size and diversity.

Over the past several years, a number of Internet information discovery tools and search engines have been introduced, including Lycos, Alta Vista, Infoseek, Yahoo and Harvest [AltaVista, Bowman94, Hardy95]. These automated tools bear most of the responsibility for organizing information on the Internet by automatically classifying and indexing collections of digital data. They periodically dispatch indexing engines, frequently referred to as "Web Crawlers", to download and examine documents in order to extract information that describes these documents [Cheong96]. The extracted information is stored in the search engine's database, along with the uniform resource locator of the site where the document resides. The stored information is later used to produce a list of resources in response to requests submitted by Internet users to query the search engine's database [Dempsey96, Hardy95, Lagoze96].

These tools have become quite popular and are helping to address the problems of resource indexing and retrieval on the Internet. Yet it has become clear that they are not sufficient to support the ever evolving information infrastructure, characterized by enormous data volume, rapid growth in the user base, and burgeoning data diversity. This is mostly due to the fact that these tools are not yet capable of dealing with a fundamental problem in information fusion – relevant documents are seldom found directly, but are discovered indirectly through other documents. As a result, queries issued by Internet users often produce an overwhelming number of responses, which frequently contain references to irrelevant material while leaving out more relevant ones.

In general, information fusion can be defined as the spatial/temporal integration of consistent as well as inconsistent information of different media and modalities from various sources. A specific example is a scientist working in a knowledge domain. This scientist would like to make his or her research

results known to scientists with similar research interests and at the same time be informed of results of interest to him or her. On the one hand this busy scientist does not want to be overwhelmed by e-mail messages produced through "push" technologies, but on the other hand this scientist is highly interested in documents which can corroborate or refute his or her research findings. If a finding can be corroborated by a document from an independent source, this scientist would like to be informed immediately. Less important documents should be left for the scientist to browse through at a convenient time. Thus an active document should have different degree of activeness. A uniform "push" technology will not be welcomed by this busy scientist.

This paper attempts to address some of the problems involved with multimedia information fusion in the Internet. Our approach is based on the concept of **active documents advertising**, whereby the metadata of a document travels in the network to seek out documents of interest to the parent document, and at the same time advertises its parent document to other interested documents. This abstraction of metadata is called an **adlet**, which is the core of our approach. In other words, it is the adlet which travels, and perform information fusion and reporting, rather than the document or the end user. Being lightweight and adaptive, the adlet serves as a surrogate of the document and indirectly as a surrogate of the end user.

An adlet's characterization of a document may range from a title or type, to key features of the document or of the author. This metadata is derived from the author's annotations of the document and is used as basis for indexing, document retrieval and information fusion. The adlet also has an advertisement strategy which defines how this active document advertises itself to the rest of the Internet. An aggressive strategy dictates the adlet to travel to other sites to seek out information, whereas a reactive strategy allows an adlet to remain at the site of creation. An adlet also has an implicit recruiting strategy derived from the metadata which determines the type of adlets allowed to become associated and/or fused with this active adlet in other to collect information for the parent document.

This paper presents the formal definition of adlets, describes the adlet system architecture and reports on an expermental prototype. The paper is organized as follows. Section 2 discusses related work. The

basic definitions for adlet are introduced in Section 3. Section 4 presents the system architecture. The visual specification of adlets is described in Section 5. Adlet composition and management are discussed in Section 6. The construction and update of virtual graph, which together with the adlets form the knowledge base, is described in Section 7. An experimental prototype based upon the active index system is being built, as described in Section 8. Concluding remarks about further research are given in Section 9.

2 Related Work

One of the recent development to overcome the limitations of Internet browsers and improve Web-browser technology aimed at adding capabilities to enable communication between HTTP servers and clients and provide interactive construction and delivery of images in response to user input.

The main idea is based on the observation that browsers have always been driven by user input. Two complimentary mechanisms, server push and client pull, were proposed to provide the server with the added capability to push new data down to the browser and the client to either reload the current data or obtain new data. A related research effort aimed at developing active objects which make it possible for a browser to download a program, execute it and display the program user's interface in a web page. This paradigm of active objects was pioneered by Sun's Hotjava browser with Java Aplets, but it needs to be further extended in the development of distributed active objects that can communicate with other active objects located on different machines across the Internet. Java aglets [Lange98] is a step in this direction.

Automated programs such as Web crawlers or indexing robots generally do not have the capability to identify the main characteristics of a document, for the Web documents lack the structure necessary to reliably extract the routine information that may result from a simple cursory inspection performed by a human user on a library index. Several approaches have attempted to address this problem and develop more efficient automated classification methods. The most common approaches seek to attach metadata to files to facilitate the indexing and classification of Internet documents based on the collected

metadata [Weibel95]. The most significant of these efforts is the Dublin Core Metadata program and the affiliated endeavor, the Warwick Framework [Dempsey96, Lagoze96]. Other research efforts such as WebSEEk [Beigi98] suggest possibilities for the indexing of visual information and demonstrate tools to combine key-word indexing with image analysis. The problem that a Web crawler may overload the server when indexing information is sought, is partially addressed by Harvest [Bowman94, Bowman95], which provides an efficient architecture based on information gatherers to reduce the amount of overhead caused by information exchange between crawlers and visited sites.

A different approach to provide better support for structuring and management of Internet information aims at developing a set of tools and languages based on a data model which is used to describe the scheme of a Web hypertext in the spirit of databases [Abit97]. Using the data model, tools are developed to support database views of the Web. These views can then be analyzed and integrated using common database techniques to generate hypertextual views of the Web [Isako95, Isako97]. A class of these research efforts view the Web as a large graph of unstructured documents and provide support for queries based on the structure of the graph. Several query systems for unstructured data have been proposed. W3QS [Kono95] uses common information retrieval techniques and the organizational properties of the hypertext to allow for both structure specifying queries and content queries, and provide management tools for Web forms. WebSQL [Mend96], WebLog [Laks96], Lorel [Abit96] and unQL [Bune96] are yet other efforts toward the formalization of languages, to support query locality, restructuring, querying heterogeneous and semi-structured information, and the provision of proper database techniques and constructs to analyze HTML documents and extract their structure.

Other approaches, such as TSIMMIS [Chawa94], aim at integrating Internet information from the Web by extracting data from heterogeneous sources and correlating these data to generate an integrated database representation of the information. Information Manifold [Levy96] provides a specific support for querying on the basis of declarative descriptions of the contents of database records accessible through a fill-in form-based interface. These approaches demonstrate the feasibility of manipulating and integrating unstructured Internet information. The next logical step is to express meta-content using XML for direct

visualization [Guha97].

The adlet system takes a different approach than most of the research efforts described above. The basic tenet of the adlet system is that in many cases it is almost impossible for a user browsing the Internet to specify accurately the type of information sought or prescribe the best action to obtain the desired documents, therefore active document abstractions (i.e., adlets) must seek out active document abstractions themselves. Furthermore, the concept of advertising among adlets avoids the necessity to export the entire contents of a given site across the network to make up an index. The capability of the adlets to join other adlets provides the basis for the development of a dynamic and distributed graph of related documents that can be retrieved on demand. In other words, adlets are specially designed for information fusion, and that is what distinguishes it from other approaches.

3 Adlets

An adlet is the means by which an active document makes itself known to other active documents. The documents are active in the sense that each document can perform certain actions with autonomy. An active document itself may post an adlet, or a user may post an adlet for an active document. An adlet has an identifier, a type, a profile, a target set of documents, a non-target set of documents, an advertising strategy and additional properties.

Definition 3.1 *An adlet is defined as: adlet = (doc-id, type, profile, target, non-target, ad-strategy, prop) where doc-id identifies the document to be advertised, type is the adlet type, profile is a set of conceptual relations from a* **concept space** *characterizing this document, target is a set of conceptual relations characterizing documents to be recruited, non-target is a set of conceptual relations characterizing documents to be avoided, ad-strategy is the advertising strategy, and prop are other properties.*

The profile can be viewed as a set of metadata elements that describe documents and user interests. Defining the elements that are necessary for describing active documents, regardless of their type or internal complexity, is a difficult task. The focus of the adlet system, however, is limited to active

document advertising, discovery and dissemination. Accordingly, the focus of the metadata for active document has been on a specific set of metadata attributes suitable for use by adlets to advertise, discover and disseminate active documents. The set of attributes must be sufficiently simple to be understood, used and possibly automatically produced by resource tools on the Internet. Furthermore, the elements of the metadata set should be extensible, and their semantics modifiable so as to allow: (i) embedding of external metadata which may be produced by an entity other than the owner or manager of the active document; (ii) refinement of the semantics of these elements into more detailed or specific semantics. Consequently, the profile of an active document in its simplest form can be specified by a set of keywords associated with the document. In general, however, the profile can be constructed as a set of conceptual relations derived from the user's annotations. (We regard a concept as a special case of a conceptual relation with a null second part.) Common conceptual relations include document type such as ('type', 'image'), ('type, 'text'), etc. and document characteristics such as ('can-be-used-as', 'genome-map'), ('schema-of', 'relational-database'), etc. Thus a profile may consist of keywords, document type and document characteristics. More complex user profiles may involve more complicated conceptual relations.

The adlet has a target set of documents, which can be defined based upon the notion of document hierarchy, i.e., documents satisfying certain conceptual relations are regarded as the target. Likewise, documents to be avoided constitute the non-target for this adlet.

The target specifies the scope and the focus of the advertisement. The **scope** of an advertisement may be global, restricted or local. Within a target the **focus** of the advertisement may be all adlets or a selected subset of adlets. The non-target specifies what is out of the scope.

The ad-strategy determines the advertising strategy used for document advertisement. The strategy could be **aggressive** or **reactive**. An aggressive strategy initiates advertisement immediately after its creation and continues advertising in a periodic fashion. A reactive strategy on the other hand engages in advertisement only in response to advertisement received from other active documents.

Example 1: A document can be a WWW page and therefore doc-id = URL. The document is characterized by a set of keywords and therefore profile = $\{keyword_1, ..., keyword_n\}$. This WWW page is in the class of browsable documents and the intended target is also a selected collection of browsable documents, i.e., target = {'Browser'}, where 'Browser' characterizes a document class in the class hierarchy. The non-target is unspecified and therefore the default is the empty set. The advertising strategy is reactive.

Example 2: A document can be a database record. The record_id is its doc-id. The document is characterized by a conceptual schema describing the conceptual structure in which this record instance is situated, and therefore its profile = { schema_name }. The target is { 'financial record' }, denoting the collection of financial records of an enterprise. The non-target is { 'scientific' }. The advertising strategy is aggressive.

Example 3: An active document may post an adlet by itself. In this case, ad = (doc-id, type, profile, {'all'}, {'none'}, ad-strategy, prop). In other words, this adlet is posted by one active document for 'all', with 'none' to be avoided. Such an adlet needs to be refined so that it can aim at more specific targets.

Definition 3.2 *Two adlets ad_1 and ad_2 can be partially ordered as follows, $ad_1 < ad_2$ if and only if the following holds:*

1. $doc - id_1 = doc - id_2$,
2. $type_1 = type_2$,
3. $profile_1 = profile_2$,
4. $target_1$ *is contained in* $target_2$, *item* $non - target_1$ *contains* $non - target_2$, *item* $ad - strategy_1 = ad - strategy_2$, *and*
5. $prop_1 = prop_2$.

In other words ad_1 is intended for a smaller target group, so ad_1 is more *refined* than ad_2.

Upon its creation, an adlet attempts to join a group of adlets, possibly creating a new one if such a group does not yet exist. Other adlets may join the group, leave the group, advertise to the group and receive advertisement from other members of the group. Adlets can negotiate among themselves so that certain goal can be reached, resulting in an adlet joining a group of adlets, advertising to the group, or receiving advertisement from the group.

Definition 3.3 *A negotiation protocol is defined as a sequence* $e{-}proc = ((d_{11}, d_{12}), (d_{21}, d_{22}), .., (d_{n1}, d_{n2}))$, *where each* (d_{i1}, d_{i2}) *is an exchanged pair of adlets and the final pair* (d_{n1}, d_{n2}) *is a pair of goal adlets.*

Example 4: Let the first entry be the consumer's and the second the producer's, then an information exchange negotiation protocol may typically look like: (('Mars',-), (-,'Heaven'), ('Sojourner', -), (-, doc-13), ('Thanks', -)).

Therefore, the adlets should induce a sequence of information exchanges e-seq between the consumer and the producer, such that $e - seq$ contains $e - proc$ as a sub-sequence. If this is feasible, then we say the negotiation protocol is supportable.

Definition 3.4 *A negotiation protocol is supportable if the adlets can induce a sequence of information exchanges e-seq such that e-seq contains* $e - proc$ *as a subsequence and the final pair* (d_{n1}, d_{n2}) *of e-seq is a pair of goal adlets.*

Additional constraints may be imposed on the supportable negotiation protocol, such as the progressive refinement of the adlets, leading to a more precise search criterion and consequently a smaller set of matching documents.

4 Adlet System Architecture

The adlet system encompasses two primary collections of objects: (i) the users with their specialized interests and the degree of interest in each type of document; (ii) the active documents to be disseminated correctly to those users with potential interest. The proper construction and maintenance of the adlet system uses the following constructs:

- Each user has a profile which can be viewed as a document summarizing the topic of interests as well as the level of interest attached to each topic.

- Every document is associated with a set of metadata elements which provide a description of the content of the document and the strategy used in advertising or recruiting a document. The metadata is used for indexing, searching, and automated processing to potentially infer and extract semantics that are specific to a particular interest group or application,

- Tools to verify potential similarities between the elements of the metadata set and user profiles. These tools can be simply provided by domain experts, such as a webster dictionary, or automatically inferred and learned based on previous handling of similar documents and user profiles. Facilities which provide these means for creating and maintaining a domain specific vocabulary currently exist [Fini94, Hais91, Pati92]. Similarity ontologies can then be used to compute the most specific generalization between the elements of a user profile or metadata set.

The system architecture to support advertising and discovery of active documents should address the issue of how to organize the resource space flexibly and dynamically in order to make feasible the implementation of a large-scaled shared interaction infrastructure, which provides adlets with efficient mechanisms for collecting, indexing, and disseminating active documents in a collaborative fashion. The architecture must:

- allow the active document structure to evolve in accordance with usage patterns, based on the advertising and recruiting strategies selected by the users,

- capture the interactions between adlets and promote advertising, filtering, archiving, and correlating documents in user and context sensitive subgroupings so as to reduce the amount of information exchanged and limit the exchange to particular groups of interested users. Capabilities must be provided to allow the adlets to organize and search the resource space by constructing links among themselves based on the metadata that describe the content and type of the document being sought

or advertised, the advertising and recruiting strategies selected by the user, and the context within which this task is being undertaken, and

- support negotiation among adlets so that every document of relevant information is ultimately advertised to every user who potentially may have interest in its content, making the aggregated collection of related documents readily available to the interested user.

To achieve these objectives, the adlet system architecture draws upon existing components of an internetwork, including name servers, authentication servers and routers, and adds new components, namely *Active Index Manager*, *Adlet Visualizer*, *Adlet Manager*, *Adlet Negotiator*, and *Adlet Communication Server*. The overall structure and functional level decomposition of the proposed architecture is depicted in Figure 1. The Adlet Visualizer supports visual specification and visualization of adlets. It allows the user to visually specify the types and properties of the adlets. The Adlet Manager supports the creation and composition of adlets. The Adlet Negotiator handles negotiation of document advertising and dissemination of active documents among adlets based on the advertising strategies and plans. These three components can be implemented as index cells of an active index (see Section 8). The Adlet Communication Server handles the exchange of adlets in a distributed system environment and can be implemented in CORBA (see Section 8).

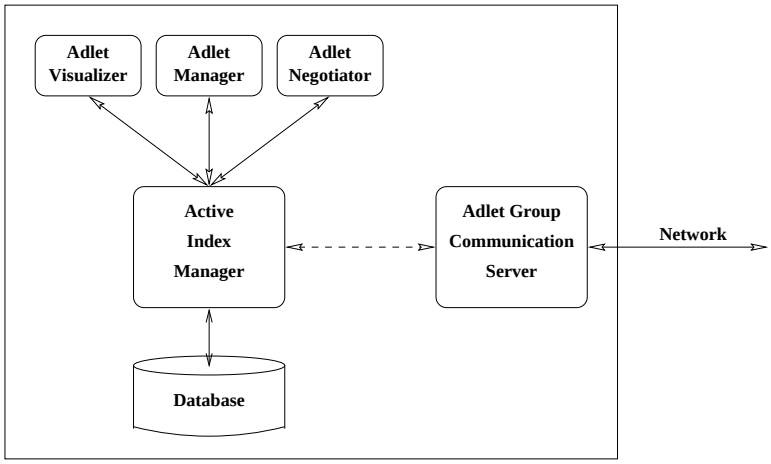

Figure 1: System architecture to support active document advertising.

The adlet system uses the metadata to organize and classify adlets with similar interests into semantically related groups. Adlets in the same group collaborate to advertise and discover new documents of potential interest to their related users.

Adlets move from site to site in order to seek adlets representing other active documents. In the process, adlets announce their existence to other adlets, advertise their metadata to and collect metadata information from other adlets residing on the visited sites. As they travel, adlets behave like "magnets". An adlet can attract other adlets and pull them together, if enough similarity exists between their metadata and if the user selected advertising strategy allows such an attraction to take place. Conversely an adlet can repulse other adlets and push them apart. The level of attraction/repulsion exerted by one adlet over another adlet determines the travel direction of an adlet.

A collection of adlets is called a **virtual site**. Virtual sites are characterized by their logical locations in some virtual space which may not necessarily correspond to a specific physical location in the system. One virtual site may span several real sites, and one real site may contain multiple virtual sites.

Each virtual site is associated with a virtual graph which represents a set of links and conceptual relations that characterize that virtual site and its relations to other virtual sites. Each link is associated with a distance. The distance between two virtual sites is proportional to the force of mutual attraction between the two sites. This distance can be selected by the user initially, and computed by the adlet system subsequently.

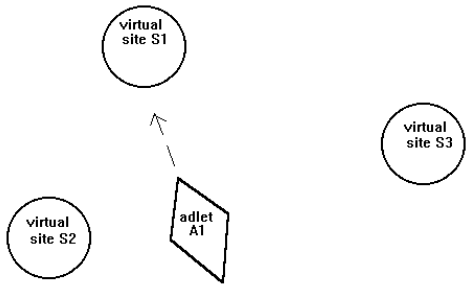

Figure 2: Visualization of adlets and sites.

Figure 2 illustrates the visualization of adlets and sites, where an adlet is represented by a wedge-shaped object and the travel direction of an adlet is indicated by the direction of the wedge's axis. The

sites are represented by round-shaped objects.

As indicated in Figure 2 an adlet does not visit all the sites on the Web, as such an itinerary involves inevitably a prohibitive number of sites. Rather, only sites which are part of the virtual graph, associated with the site where the adlet is created, are visited. Since this virtual graph is dynamically updated, when new adlets are created and mutually attractive adlets are merged, the adlet's effective sphere of motion continuously changes over time.

In what follows, we first begin by discussing visual specifications of adlets, and then proceed to discuss the adlet composition and management functionalities.

5 Visual Specification of Adlets

When a document is created, the user can visually specify the associated adlet. The design of the adlet visual specification language is based upon the theory of visual languages [Chang96a]. The mode of interaction is direct manipulation. The user annotates the objects on the screen by pointing, clicking and entering keywords. The user first defines a **concept space** where the concepts are color-coded. Each adlet type has a dominant color which is the concept that characterizes it most closely, and other less dominant colors corresponding to the concepts that characterize it with varying degree of closeness. The darkness or lightness of a color indicates the closeness of this characterization. Each adlet type also has a dominant color which is the concept that attracts it most, and other less dominant colors for less attractive concepts. Finally, concepts that repulse the adlet are also defined. These color codes are shown in different areas on the adlet or dynamically shown in an alternating fashion.

Figure 3 illustrates the visual specification of an adlet which carries the following information:

"I represent doc, I am profile, I love target, I hate non-target, I stay put (reactive) or travel (aggressive) and I behave like prop".

The **attraction/repulsion** property of an adlet is important in determining the motion behavior of an adlet. An attraction/repulsion force of an adlet to a color-coded concept can be one of the following, as specified by the user (perhaps by clicking on a sliding scale):

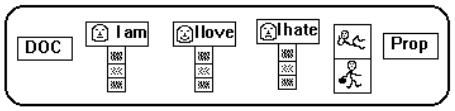

Figure 3: Visual specification of an adlet.

A1. *Fatal attraction*: the adlet must merge with an adlet/site of said color.

A2. *Strong attraction*: the adlet attempts to merge with an adlet/site of said color until precondition becomes 'false'.

A3. *Weak attraction*: the adlet makes a best effort attempt to merge with an adlet/site of said color.

A4. *Neutral*: no attraction/repulsion is allowed.

A5. *Weak repulsion*: the adlet makes a best effort attempt to avoid an adlet/site of said color.

A6. *Strong repulsion*: the adlet attempts to avoid an adlet/site of said color until precondition becomes 'false'.

A7. *Max repulsion*: the adlet avoids the adlet/site with said color, even if this means its own destruction.

The ad-strategy of an adlet type can be selected by clicking on either "reactive" or "aggressive". Other properties of an adlet type to be specified by the user include the following:

B. *Destructability*: an adlet can either be destructive or nondestructive. A destructive adlet vanishes when a specific condition becomes 'true'. Nondestructive adlet cannot be destroyed under ordinary circumstances.

C. *Regenerability*: an adlet can be regenerating or nonregenerating. A regenerating adlet is able to create identical instances of its own type. Nonregenerating adlets do not possess this capability.

D. *Migration*: an adlet can be migrating or nonmigrating. A migrating adlet seeks out and migrate to sites where other adlets of similar interests reside. Nonmigrating adlets remain on the site

where they were created. The *adstrategy*, either aggressive or/reactive, leads to either migrating or nonmigrating adlets, respectively.

E. *Temporal Sensitivity*: an adlet can be temporally sensitive or temporally insensitive, depending on whether its predicates and actions involve time as a parameter.

F. *Location Sensitivity*: an adlet can be location sensitive or location insensitive, depending on whether its predicates and actions involve location as a parameter.

G. *Context Sensitivity*: Adlet can be context sensitive or context insensitive, depending on whether its predicates and actions involve contextual variables as parameters.

After an initial session to perform a detailed annotation of an adlet description, the user can create an adlet type which can serve as a template for future similar document types.

6 Adlet Composition and Management

When adlets in the same virtual site collide, meaning that their mutual attraction force exceeds a certain threshold, the **composition rules** dictate whether the two adlets should be merged to form a new adlet, with the possible side effect of updating the virtual graph. Therefore, through adlet interaction and composition an active document can be linked to other active documents, thereby leading to the discovery of new relationships and information fusion. The adlet operators that allow the interaction and composition of adlets can be described as follows:

1. **Concatenation**: Adlet 1 and adlet 2 are combined into a train of adlets which is then regarded as a single adlet. The visual operator is a 'train'. An example of an adlet requiring the concatenation operator is an **information-gathering-adlet**, which concatenates all the matched adlets it encounters during its itinerary, and carries them back to the parent document or the end user.

2. **Equi-Join**: Adlet 1 and adlet 2 are joined into a single new adlet. The properties of the new adlet are inherited from both parents. The visual operator is a 'heart'. An example of an adlet requiring

the equi-join operator is an **archive-building-adlet**, which finds the common features of related documents so that later on it can create a virtual site and links to these documents.

3. **BW-Join**: Adlet 1 destroys adlet 2 but takes over some of the latter's properties. The visual operator is a 'spider' – the Black Widow. This operator is applicable only when adlet 2 is a copy of adlet 1, so that adlet 1 can absorb the information gathered by adlet 2 and then adlet 2 (the copy) is destroyed. An example is a **user-profile-adlet** which broadcasts itself to many sites to collect URLs of related documents and then combine these lists for the parent document or the end user.

4. **Split**: Adlet 1 is split into adlets 2 and 3. The unary visual operator is represented by a 'pair of scissors'. The most common example is an information-gathering-adlet has gathered sufficient information, so that it is split into two, one returning to the parent document (and the end user) to report its findings, and the other continuing its Odyssey.

These operators can be combined to specify more complex operators. For example two adlets can first be merged by BW-join and the new adlet is then split into two. Thus we can specify adlets that are: consumed, forever circulating, procreative, and so on.

The operators are applied when the adlets are combined. Therefore the application sequence is time-dependent. The application of any of the above operators may lead to the update of the virtual graph, because when the adlets are merged, the documents they refer to need also to be linked to each other. To reflect these new associations between documents, dynamic links are added to the active documents and directed arcs are added to the underlying virtual graph.

The effect of the interaction among adlets is a continuous, dynamic update of the virtual graph. A site's virtual graph and its adlets together form its knowledge base. Therefore an important issue is when and how to update the virtual graph of a site.

7 Virtual Graph Construction and Update

Active documents in our approach evolve under constraints so that the system can improve over time. This evolution is made possible due to the interaction among adlets, the recruiting and establishments of groups, the dynamic linking of documents and the updating of virtual graphs. Since the knowledge base consists of the adlets and their virtual graph, an important issue in making this constrained evolution possible relates to when and how to update the virtual graph of a site. Every time a new adlet is created, either because a new document is created or because two adlets are merged, a decision must be made regarding the update of the virtual graph of the active document responsible for generating this adlet. In the following subsections, we first provide a formal definition of a virtual graph. We then proceed to discuss the data structure used to represent the virtual graph, and describe the basic steps used by a node the to update the virtual graph in response to adlet interactions. We finally describe how the virtual graph is used by a migrating adlet to select the virtual sites to be visited during its itinerary.

7.1 Virtual Graph Specification

A virtual graph, VG, can be formally defined as a set of nodes $N = \{n_1, ..., n_k\}$ and a set of arcs $A = \{a_1, ..., a_n\}$. Each node $n_i = (S_i, T_i)$ is characterized by a site S_i and an adlet tree T_i. An arc (n_i, n_j) connects node n_i to node n_j if the two sites S_i and S_j are conceptually related. Initially the virtual graph contains only a single node n_i for the local site S_i. Assume site S_i, has a set of documents $D_i = \{doc_1, ..., doc_k\}$. The adlets, $adlet_1, ..., adlet_k$, corresponding to each document of D_i respectively, are then organized into a tree, T_i. The tree, T_i, represents a conceptual hierarchy which reflects the similarity between the parent documents of the adlets. hierarchies of adlets in a node are dynamically updated as a new adlets are added to the node or an existing adlets are deleted from the node.

A virtual graph, VG, can be viewed as a set of embedded hierarchies of adlets. A subgraph of VG represents a conceptual view of the collection of information in a virtual site. These hierarchies and conceptual views facilitate graph processing and searching. Adlets may carry time-stamps to facilitate conceptual view integration through adlet merging and virtual graph updating. An example of the

embedded hierarchies and conceptual views is illustrated in Figure 4.

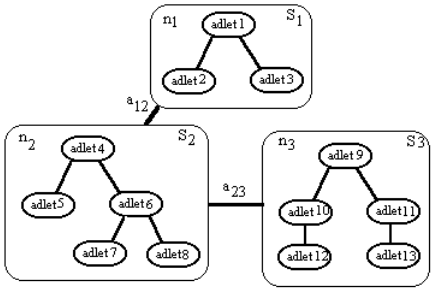

Figure 4: A virtual graph with embedded hierarchies of adlets.

In the virtual graph, each node has its own information such as how many and what kind of adlets reside in a particular site. In addition to its local adlet information, a node has information about adlets which resides in other nodes. This information include the distances between virtual sites, which as specified earlier are proportional to the force of mutual attractions between these sites. Distances for conceptually related nodes can be expressed in terms of typical documents distances, such center document distances in document clusters or or average-like distances of related documents between two nodes [Chang98].

The distance information of adlets that reside on other nodes can be shared among local adlets. Sharing the distance information leads to low memory usage and reduces the effective computation required to retrieve related documents. The distance, along with the advertising strategy, are used by an adlet to determine whether a site is worth visiting.

Based on the distance information and the underlying advertising strategy, the adlet selects a set virtual sites to be visited. As it travels, the adlet collects information about similar documents which can later be used to narrow down the set of documents which satisfy the user specified conditions before retrieving those documents that are of interest to the user. Consequently, when a user poses a query involving a specific document and specifies "how far" the search for related documents can proceed, the related adlet uses the distance information and information about other documents, gathered during previous itineraries, to determine which documents must be retrieved to satisfy the user's query.

7.2 Virtual Graph Representation

In order to dynamically update the virtual graph in response to the creation of new adlets or the merging of mutually attractive existing adlets, the node maintains a *node-information-table* for each node it is connected, along with the distance to that node. Each row of the node-information-table has the the following format:

type name distance route doc-id profile target non-target ad-strategy

The attributes associated with the rows of a node-information-table row have the following meanings:

- *type*: this attribute represents a tag string to distinguish adlet information from other information and characterize its type.
- *name*: this attribute represents the name of the adlet.
- *distance*: this attribute represents the distance to reach this adlet from this local node. If this adlet resides on the local node, the distance is 0 (zero).
- *route*: this attribute represents the path to be followed starting from this node in order to reach the adlet specified in the *name* attribute. If the adlet resides on this local node, the value is *null*.
- *doc-id*: this attribute represents the identity of the document represented by the adlet specified in the *name* attribute. The *doc-id* attribute uniquely identifies the document to be advertised.
- *profile*: this attribute represents the profile of the document represented by the adlet specified in the *name* attribute. A profile is a set of conceptual relations from a concept space characterizing the document specified in the *name* attribute.
- *target*: a set of conceptual relations characterizing documents to be recruited.
- *non-target*: a set of conceptual relations characterizing documents to be avoided.
- *ad-strategy*: advertising strategy for the adlet specified in the *type* attribute.

The attributes for an adlet such as *distance*, profile, *target*, and *non-target* can also be pointers to other data structures or database records.

To illustrate an instance of this data structure consider the virtual graph depicted in Figure 5. The virtual graph consists of a set of nodes $N = n1, n2, n3, n4, n5$. Each link of the graph is labeled with the distance between the virtual sites associated with the nodes of the virtual graph. In this case, the distance between nodes $n1$ and $n2$ is 5.

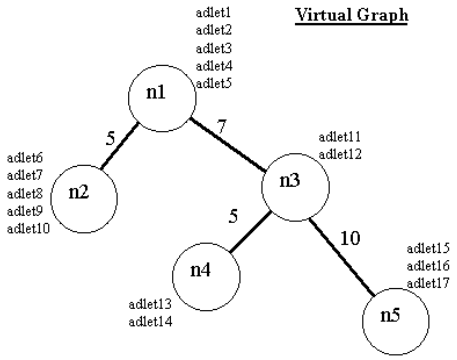

Figure 5: An example of virtual graph.

The node-information-table associated with node $n2$ may initially contain the following information.

nodename n2

connectednode n1 5

type adlet6 0 null doc6 profile6 t6 nont6 aggressive

type adlet7 0 null doc7 profile7 t7 nont7 aggressive

type adlet8 0 null doc8 profile8 t8 nont8 aggressive

type adlet9 0 null doc9 profile9 t9 nont9 aggressive

type adlet10 0 null doc10 profile10 t9 nont9 progressive

The first row of the node-information-table identifies $n2$ as the owner of the table. The information in the second row, connectednode n1 5, states that $n2$ is connected to $n1$ and the distance between the two nodes is 5. The information contained in the third through seventh rows describe the attributes of the adlets that currently reside on node $n2$.

When node $n2$ is added to the virtual graph VG of node $n1$, $n2$ sends its node-information-table to node $n1$. Node $n1$, which already has included nodes $n3$, $n4$, and $n5$ in its own information table uses the information contained in the node-information-table of node $n2$ to update its own table and reflect the addition to the virtual sites of $adlet6$, $adlet7$, $adlet8$, $adlet9$, and $adlet10$. When the update is completed, node $n1$ sends its updated node-information-table to node $n2$ which uses this new information to update its own table.

The node-information-table remains the same until new adlets are created or existing adlets are merged. The basic steps of the procedure used by nodes to update their node-information-table, in response to adlet creations and deletions, are described below.

7.3 Virtual Graph Update Algorithm

As stated above, the virtual graph update algorithm is invoked when a change, such as addition or a deletion of an adlet. occurs in a node. The latter notifies the other nodes that such a change has occurred. Furthermore, since the arcs connecting nodes represent conceptual relations between the nodes at each of the arch, the node where a change has occurred sends its updating information only to those nodes that are directly connected to it.

When a node receives updating information, it validates the received information, compares and eventually updates its own node-information-table with the received information. Upon updating its own table, the node then sends the resulting node-information-table to all of its connected nodes, except the one from which the update information was received. Eventually, the updated information propagates through the entire virtual graph. This updating algorithm can be summarized as follows:

Step 1: When a change occurs, the node (sender) updates its own node-information-table and sends the updated information to each node directly connected to it (receivers).

Step 2: The receiver compares the content of the newly received node-information-table with the content of its own node-information-table. If a difference between the contents of the two node-information-tables is observed, the receiver updates its node-information-table to reflect the new changes as follows:

- If the sender is a new node, the receiver first adds an entry to its own node-information-table which contains the name of the sender and the distance as specified in the sender's node-information-table.
- If a new adlet is specified in the update information, the receiver adds the name of the adlet and the distance to its own node-information-table. The distance is obtained by adding the distance between the sender and receiver to the distance of the adlet. The route for the new adlet is set to the name of the sender.
- If the sender is not new, and the receiver determines that a previously known adlet, which is currently part of the receiver's own node-information-table, is not contained in the update information, the known adlet is considered no longer part of the virtual graph. The receiver removes the entry from its own node-information-table.

Step 3: If the receiver's set of directly connected nodes contains no node other than the original sender of the update information, the node-information-table update procedure completes. On the other hand, if the set contains other than the original sender, the receiver becomes a sender and sends the newly updates node-information-table to all its directly connected neighbors. Furthermore, to avoid sending information back and forth between nodes indefinitely, a logical timestamp, which uniquely identifies the original sender and the latest version of the update information sent, is used with each message. The logical timestamp prevents any node from sending the same update information to the node where this information originated from.

These steps are progressively executed by all nodes in the virtual graph. Eventually, the update information is disseminated to all nodes members of the virtual graph and the conceptual views (i.e., the subgraphs of the virtual graph) of the virtual site are consolidated.

7.4 Migration of Adlets

When a new document is created, if the corresponding adlet is reactive, the adlet stays in the local site and nothing more happens. If this adlet is aggressive, the adlet travels to a dynamically computed **multicast set** of virtual sites, according to the following policy: If (n_i, n_j) is an arc in the virtual graph,

we compute the conceptual distance between S_i and S_j, and if the distance is less than a predetermined threshold then S_js is added to the multicast.

The addition of new sites proceeds in the increasing order of the conceptual distances associated with the sites. In other words S_j with the smallest conceptual distance is added to the multicast set first, followed by the one with the next smallest conceptual distance and so on, until the preset maximal size of the multicast set is reached. The adlet can then travel to the nodes in the multicast as specified by the underlying advertising strategy and negotiation protocol.

8 An Experimental Prototype

In our implementation of the experimental prototype we employ the Active Index Manager to manage adlets and active documents. Our approach is that an adlet is instantiated through the instantiation of active index cells [Chang95]. Using this approach the Active Index Manager will manage all adlets (index cells) and handles adlet migration through distribution of the index cells. The Adlet Visualizer, Adlet Manager and Adlet Negotiator are also implemented as special active index cells.

We are currently building an experimental prototype based upon the active index approach. VisiBroker 3.0 is used to implement the Adlet Communication Server to handle the exchange of adlets in a distributed system environment.

In what follows we give an example of adlets in action. The behavior of adlets as index cells in the active index is described in parentheses.

8.1. Initial State: As shown in Figure 4, the virtual graph VG has three nodes n1, n2 and n3. Node n1 is for site S1 with adlets adlet1, adlet2 and adlet3. Node n2 is for site S2 with adlets adlet4, adlet5, adlet6, adlet7 and adlet8. Node n3 is for site S3 with adlets adlet9, adlet10, adlet11, adlet12 and adlet13. There are arcs a12 and a23, indicating S1 and S2 are conceptually related, and S2 and S3 are conceptually related.

8.2. Adlet Creation: Suppose document3 has just been added to site S1, and the corresponding

adlet3 is added to the virtual graph. If document3 is the user profile, then the added adlet3 is the user-profile-adlet. (In active index, this means an index cell ic3 is added to site S1.)

8.3. Adlet Migration: If adlet3 is aggressive, it will travel to conceptually related nodes. Since there is an arc a12, adlet3 can travel to site S2. (In active index, this means index cell ic3 will send a message to Active Index Manager of site S2 to clone itself. But the original ic3 will stay at site S1. In other words, the original adlet3 will never leave its home site. It is its clone that will materialize in other sites.)

8.4. Determination of Matched Adlets: Adlet3, once at site S2, will try to merge with adlets there. This is done by finding whether its target matches another adlet's profile, and its non-target does not match another adlet's profile. Let us say adlets adlet5 and adlet7 are thus identified. (In active index, the cloned ic3 will broadcast messages to all ic's at site S2 and ic5 and adlet7 will respond.)

8.5. Adlets Merging by BW-Join: Since adlet3 is a user-profile-adlet, therefore adlet3 and adlet5 are merged. The adlet composition operator is the BW operator, symbolized by the Spider. Adlet3 absorbs adlet5. This means document3 should absorb some of the information contained in document5. What happens is that updated adlet3 will travel back to site S1 to update original adlet3. (In active index, the message sent by ic5 to ic3, will cause ic3 to send a message back to the original ic3 at site S1, causing it to be updated.) Similarly, adlet3 and adlet7 are merged.

8.6. Updating of Virtual Graph: Using the algorithm described in the previous section, the virtual graph is updated. (In active index, updating of virtual graphs may be an action performed by a special ic.)

8.7. Retrieval of Documents: When a user wants to retrieve document3, the related documents discovered by adlet3 will be retrieved. Since adlet3 is the user-profile-adlet, the retrieved documents will be those documents conceptually related to the user profile. (The retrieval_ic is activated, which activates ic5 and ic7, to retrieve the actual documents document5 and document7.)

8.8. Establishment of New Group: After negotiations, adlet5 of document5 and adlet7 of document7 may decide to form a group with links between them. A new virtual site is thus established, with its associated new virtual graph and adlets.

Figure 6: Home page of experimental adlet system.

An experimental adlet system is being designed for information retrieval and fusion applications in the NSF Plant Genome Research Program which has an objective to develop shared resources and research tools that will enable plant genome research to advance efficiently, rapidly and in a cost-effective manner. As shown in Figure 6, the experimental plant genome adlet system will be accessible through the Internet using a browser. The users - scientists, educators and students - can create annotated active tags (i.e. adlets) for genome data documents. The web page to create such adlets is illustrated in Figure 7. The contents of two adlets are shown below, the first adlet for a note describing an experimental design, and the second for a graph on Collinsia verna germination in 1977.

Adlet 1

I-represent = Collinsia verna germination 1997

I-am = Differences Among Populations, Germination Schedules

I-love = Germination Schedules

I-hate = Tillage Practices, Herbacide Schedules

I-behave = aggressive

I-keyword = germination, Collinsia verna

I-comment = Fall germination for seeds from four populations of Collinsia verna grown reciprocally at

the four sites in western Pennsylvania.

Adlet 2

I-represent = Maternal Effects Experimental Design

I-am = Levels of Selection, Family Variation, Maternal Effects

I-love = Germination Schedules

I-hate = Tillage Practices, Herbacide Schedules

I-behave = aggressive

I-keyword = Collinsia verna, Phacelia purshii

I-comment = Experimental design for an experiment looking at the importance of seed dormancy and maternal effects.

These adlets are active and can travel from node to node, collecting additional information. The adlets can be organized hierarchically so that only adlets in a certain class are active during a user's search. For example an expert may want to retrieve information annotated by fellow scientists only, while a student may want to retrieve information annotated by teacher, fellow students and expert scientists. The user must provide a knowledge hierarchy organized as shown in the following example, where the nested parentheses are used to indicate more specialized concepts in the hierarchy.

Dormancy (Gene (QTL Markers) Individuals (Induction, Fatal Germination, Embryogenesis, Parental Effects (Maternal Effects) Seed Envelopes)

This knowledge hierarchy is used in computing the similarity among adlets and in estimating the distance among virtual sites.

The Plant Genome Adlet System, once completed, will first be tested by researchers and graduate students of the University of Pittsburgh, and then serve as a testbed for further research and development of the proposed approach.

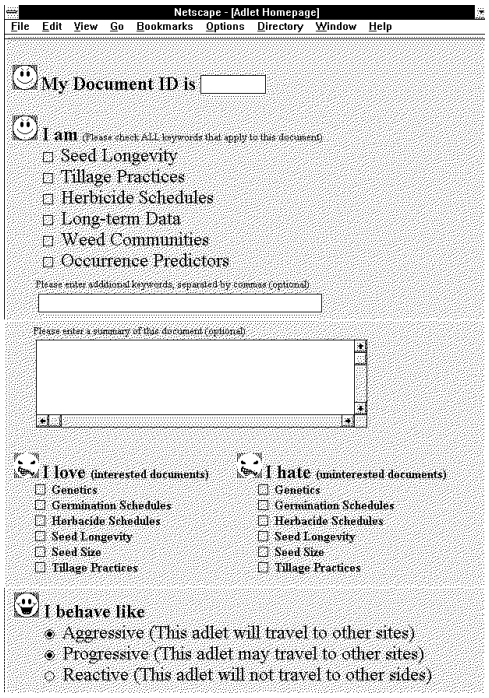

Figure 7: Web page to create adlets.

9 Discussion

Several factors make the proposed approach potentially unique and effective:

- adlets allow users to more directly and more completely reflect their personal views and concepts of the network resources, rather than those of a particular crawler or search engine.

- adlets have the ability to roam from site to site and progressively acquire new knowledge which is then used to further refine their advertising and recruiting strategies and tailor them to the needs of the user. This ability sets adlets apart from crawlers which execute remotely at different sites and collect general indexing information about documents which may not be of interest to the local users. It also provides the basis for the development of more focused, user tailored information fusion, retrieval and dispersal tools.

- adlets have the ability to execute asynchronously, process information offline and limit their interactions to within their interest groups, thereby allowing the system to scale gracefully to the addition of new users and new documents.

- adlets have the potential to reduce the large amount of communication traffic and computation overhead, typically associated with Internet crawlers and search engines, by limiting document metadata advertising and recruiting to interested groups.

The first two factors address the need of the Internet users to rapidly and efficiently acquire documents that are relevant to their interests and profiles, while last two factors reflect the need for more efficient and more scalable resource management tools which can easily cope with the addition of new users and new documents.

The proposed approach can be deployed to both nonactive and active networks as described below. Active networks allow individual user, or groups of users, to inject customized programs into the nodes of the network. We can deal with hybrid active networks where some or even all nonlocal nodes are unable to handle adlets. The approach is to provide an **adlet extractor** to extract adlets from documents. The adlet extractor can be manually, partially automatic, or fully automatic, depending upon the type of multimedia documents to be processed. At an adlet-enhanced node, incoming documents are converted into active documents by the extraction of adlets. The ad-strategy for such adlets will be reactive, i.e., these adlets that are extracted from documents will remain in their sites and do not travel. Thus adlet interaction is dealt with by the local node equipped with the adlet-savvy software. But as more and more nodes are enhanced with adlet-savvy software, the network becomes more active, and the behavior of adlets more sophisticated. Therefore, a) any document, including web page, database record, video file, audio file, image and even paper documents, can be enhanced by an adlet and become an active document; b) any node in a nonactive network can be enhanced by adlet-savvy software and the adlet-enhanced node can co-exist with other non-enhanced nodes. These two important features make the above described approach applicable to multimedia information fusion, information retrieval, data mining, geographic information systems, medical information systems and disaster management. The experimental adlet system will provide a testbed for feasibility studies in such an active or hybrid network environment.

There are many topics for further research and development, and the most important ones are briefly discussed below.

- *Adlet Visualization*: In sections 5 and 6, we described an approach for the visual specification of adlet types by annotation, and the visual operators for merging adlets. The semantics of this visual language [Chang96a] needs to be carefully investigated. Concerning visualization, we need to investigate whether text visualization [Hearst95, Rao95] and 3D visualization techniques [Robert93] are applicable, and how best to determine the direction of adlet movement.

- *Adlet Migration and Multicast Protocols*: We need to design the negotiation protocols so that given an advertisement plan the goal can be reached and the exchange of adlets and other network primitives leads to the goal of completed exchanges. In adlet interaction, who visits what site? Adlets migration may happen before or during or after adlets interaction. An important topic is how to apply multicast protocols in support of adlets negotiation protocols and routing.

- *Implementation Issues:* The current implementation strategy lends itself to software portability, so that the adlet system can easily be ported to another PC environment. Other implementation approach should be tried, including the use of aglets to implement the adlet system. In the current implementation, a PC can easily handle thousands of adlets. When there are a large number of adlets, we need to consider ways of getting rid of obsolete adlets using time-outs and other techniques. The use of *adlet extractor* and *adlet filter* may further reduce the search space for adlets and should be further investigated.

Acknowledgements

The research of the first author was supported in part by the National Science Foundation under grant IRI-9224563, "An Active Image Information System for Biomedical Image Databases", and supplementary award IIS-9841137, "A Plant Genome Information Assistant". The virtual graph updating algorithm was developed by Miyawaki Kosuke. The Adlet prototype system is being developed by Ping-Wen Chen,

Guanlin Shen and Longjiang Yang.

References

[Abit96] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wienner, "The Local Query Language for Semi-Structured Data", http://www-db.standford.edu, 1996.

[Abit97] S. Abiteboul, "Querying Semi-Structured Data", In Sixth International Conference on Data Base Theory", Lecture Notes in Computer Sciences, 1997.

[AltaVista] AltaVista Technology Home Page, http://www.altavista.com

[Beigi98] Mandis Beigi, Ana Benitez, and S.-F. Chang, "MetaSEEk: A Content-Based Meta Search Engine for Images," SPIE Conference on Storage and Retrieval for Image and Video Database, San Jose, Feb. 1998.

[Bowman94] C. Bowman, P. B. Danzig, Udi Manber, and M. F. Schwartz,"Scalable Internet Resource Discovery: Research Problems and Approaches", *Communications of the ACM*, Vol. 37, No. 8, August 1994, 98-107.

[Bowman95] C. Bowman, P. B. Danzig, Udi Manber, and M. F. Schwartz, "The Harvest Information Discovery and Access System", *Computer Networks and ISDN*, Vol. 28, Nos. 1-2, Dec 1995, 119-125.

[Bune96] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu, "A Query Language and Optimization Techniques for Unstructured Data", In ACM SIGMOD, International Conference on Management of Data, SIGMOD'96, Montreal, Canada, pp. 505-516, 1996.

[Chang95] S. K. Chang, "Towards a Theory of Active Index", Journal of Visual Languages and Computing, Vol. 6, No. 1, March 1995, 101-118.

[Chang96a] S. K. Chang, "Extending Visual Languages for Multimedia", IEEE Multimedia Magazine, Fall 1996, Vol. 3, No. 3, 18-26.

[Chang96b] S. K. Chang, "Active Index for Content-Based Medical Image Retrieval", *Journal of Computerized Medical Imaging and Graphics*, Special Issue on Medical Image Databases (S. Wong and H. K. Huang, eds.), Elsevier Science Ltd., 1996, 219-229.

[Chawa94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom, "The TSIMMIS Project: Integration of Heterogeneous Information Sources", In IPSJ Conference, Tokyo, 1994.

[Chang98] S. K. Chang, D. Graupe, K. Hasegawa and H. Kordylewski, "An Active Multimedia Information System for Information Retrieval, Discovery and Fusion", *International Journal of Software Engineering and Knowledge Engineering*, March 1998, Vol. 8, No. 1, 139-160.

[Cheong96] F. C. Cheong, "Internet Agents, Spiders, Wanderers, Brokers and Bots", *New Riders Publishing*, 1996.

[Dempsey96] L. Dempsey and S. L. Weibel, "The Warwick Metadata Workshop: A Framework For the Deployment of Resource Description", *D-lib Magazine*, July-August 1996.

[Fini94] T. Finin, R. Fritzson, D. McKay, and R. McEntire "KQML as an Agent Communication Language", Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)", ACM Press, November 1994.

[Gene92] M.R. Genesereth and R. E. Fikes, "Knowledge Interchange Format", Technical Report, Logic-92-1, Stanford University, January 1992.

[Guha97] R.V. Guha, and T. Bray, "Meta Content Framework Using XML," Technical Report NOTE-MCF-XML, W3C, 1997, http://www.w3.org/TR/NOTE-MCF-XML/.

[Hais91] L. J. Haisting, "EDI: A New Way of Doing Business", St. Paul Software, St. Paul, MN, 1993.

[Hardy95] D. R. Hardy, M. F. Schwartz, D. Wessels, "Harvest User's Manual, Version 1.2", *Technical Report CU-CS-743-94*, November 1994.

[Hearst95] M.A. Hearst, "TileBars: Visualization of Term Distribution Information in Full Text Information Access," The Proceedings of CHI'95 (Conference of Human Factors in Computing Systems: Mosaic of Creativity), ACM, 1995.

[Isako95] T. Isakowitz, E.A. Stohr, and P. Balasubramanian, "RMM: A Methodology for Structured Hypermedia Design," Communications of the ACM, vol. 38, no. 8, pp. 34-44, August 1995.

[Isako97] T. Isakowitz, A. Kamis, and M. Koufaris, "Extending the Capabilities of RMM: Russian Dolls and Hypertext," Proceedings of 30th Hawaii International Conference on System Sciences (HICSS-30), Maui, Hawaii, January 7-10, 1997.

[Kim95] W. Kim, Editor, "Modern Database Systems: The Object Model, Interoperability, and Beyond", ACM Press and Addison Wesley, 1995.

[Kono95] D. Konopnicki and O. Shmueli, "W3QS: A query system for the World-Wide Web", In Intern. Conf. on Very Large Data Bases (VLDB'95), pp- 54-65, 1995, Zurich.

[Lagoze96] Lagoze, Carl and Lynch, Clifford and Daniel, Ron, Jr. June, 1996. "The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata.", Cornell Computer Science Technical Report TR96-1593.

[Laks96] L. Lakshmann, F. Sadri, and I.N. Subramanian, "A Declarative Language for Querying and Restructuring the Web", In 6th Intern. Workshop on Research Issues in Data Engineering: Interoperability of Nontraditional Database Systems (RIDE-ND'96), 1996.

[Lange98] Danny Lange and Mitsuru Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, ISBN 0-201-32582-9, 1998.

[Levy96] A.Y. Levy, A. Rajaraman, and J.J. Ordille, "Querying Heterogeneous Information Sources Using Source Description", In Intern. Conf. On Very Large Data Base, VLDB'96, Mumbai, Bombay, 1996.

[Mend96] A. Mendelzon, G. Mihaila, and T. Milo, "Querying the World Wide Web", In the First Int. Conf. on Parallel and Distributed Information Systems (PDIS'96), 1996.

[Nech91] R. Neches, R. Fikes, P. Patel-Schneider, R. Patil, T. Senator, and W. Swartout, "Enabling Technology for Knowledge Sharing", AI Magazine, Vol.12, No. 3, September, 1991.

[Pati92] R. Patil, R. Fikes, P. Patel-Schneider, D. McKay, T. Finn, T. Gruber, and R. Neches, "The DARPA Knowledge Sharing Effort: Progress Report", Proceedings of the Third International Conference of Knowledge Representation and Reasoning", Morgan Kaufmann, San Fransico, CA, November 1992.

[Rao95] R. Rao, J.O. Pedersen, M.A. Hearst, J.D. Mackinlay, S.K. Card, L. Masinter, P.K. Halvorsen,

and G.G. Robertson, "Rich Interaction in the Digital Library," Communications of the ACM, vol. 38, no. 4, pp. 29-39, April 1995.

[Robert93] G.G. Robertson, "Information Visualization Using 3D Interactive Animation," Communications of the ACM, vol. 36, no. 4, pp. 57-71, April 1993.

[Sheth90] A. P. Sheth, and J. A. Larson, "Federal Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases" ACM Computing Surveys, Vol. 2, No. 3, September 1990, 183-236.

[Weibel95] Weibel, Stuart, "Metadata: The Foundations of Resource Description", *D-lib Magazine*, July, 1995.