

InfoHarness: Use of Automatically Generated Metadata for Search and Retrieval of Heterogeneous Information

Leon Shklar^{*+}, Amit Sheth¹, Vipul Kashyap¹⁺, and Kshitij Shah^{*+}
shklar@cs.rutgers.edu

Abstract. The *Infoharness*TM system is aimed at providing integrated and rapid access to huge amounts of heterogeneous information independent of its type, representation, and location. This is achieved by extracting metadata and associating it with the original information. The metadata extraction methods ensure rapid and largely automatic creation of information repositories. A stable hierarchy of abstract classes is proposed to organize the processing and representation needs of different kinds of information. An extensible hierarchy of terminal classes simplifies support for new information types and utilization of new indexing technologies. InfoHarness repositories may be accessed through Mosaic or any other HyperText Transfer Protocol (HITP) compliant browser.

1.0 Introduction

The accelerating explosion in the amounts and variety of information has made the knowledge about its existence, location, and the means of retrieval very confusing. The information ranges from software artifacts to engineering and financial databases, and comes in different types (e.g., source code, e-mail messages, bitmaps) and representations (e.g., plain text, binary). This information has to be accessed through a variety of vendor tools and locally developed applications.

There have been attempts to address this problem by building information repositories that depend on relocating and reformatting the original information [20]. Such approaches require the design and maintenance of ever-changing format translators. The initial conversion of information requires substantial human and computing resources. Maintaining the repositories presents the additional dilemma of either creating new and updating existing information in the uniform format, or continuously managing changing data in different formats. The latter may be partially remedied through recent efforts in the uniform representation of heterogeneous documents [29], but this does not help with arbitrarily formatted data.

Our main objective in constructing the InfoHarness system is to provide rapid access to huge amounts of heterogeneous information in a distributed environment without any relocation, restructuring, or reformatting of data. Many researchers have investigated the use of metadata to support run-time access to the original information [1,3,8,9,11,12,13,19,26]. Others [5,11,21,27] have investigated the use of data mining for the automatic extraction of metadata. We refine and synthesize some of the ideas contained in these efforts to provide advanced search and browsing capabilities without

* Bell Communications Research, 444 Hoes Lane, Piscataway, NJ 08854

¹ LSDIS, Department of Computer Science, University of Georgia, Athens, GA 30602-7404

+ Computer Science Department, Rutgers University, New Brunswick, NJ 08902

TM InfoHarness is a trademark of Bell Communications Research, Inc.

imposing constraints on information suppliers or creators. Further, we propose and develop a *stable* abstract class hierarchy, which need not be modified to define terminal classes that accommodate new types of information and new indexing technologies.

InfoHamess has been designed with an open, extensible, and modular architecture. The InfoHamess prototype is now operational and on trial at Bellcore for building repositories of heterogeneous software artifacts [25], accessing geo-spatial data, and a variety of other applications. It supports the largely automatic generation of InfoHamess repositories, and provides access to the physical information from Mosaic and other World-Wide Web (WWW [2]) browsers through an HTTP gateway. We expect to make the system available on the Internet in the first half of 1995.

The key features of InfoHamess include:

- Providing advanced search and browsing capabilities without restructuring, reformatting, or relocating information.
- An extensible terminal class hierarchy to model arbitrary formatted information.
- The ability to utilize third-party indexing technologies.
- Extensive use of metadata for mapping logical views of information to physical files.
- A high-level declarative language [26] to provide power and flexibility in controlling the metadata generation process (now under development).

In this paper, we have analyzed *content-based* and *content-descriptive* metadata and their utility in the search, retrieval and browsing of information. Content-based metadata is based on the content of documents (e.g., the document vectors in the LSI index and the complete inverted WAIS index). Content-descriptive metadata serves to describe documents and includes *domain-independent* metadata (e.g., the location and size of a document) and *domain-dependent* metadata which abstracts or attempts to capture the semantic meaning of information. We also take a look at the classification of metadata based on their utility (a modified version of the one in [3]) and see how various attempts at using metadata (including the one in InfoHamess) conform to the classification.

The organization of the paper is as follows. In section 2, we discuss the role of metadata in providing transparent access to information. In section 2.1, we discuss the organization of metadata. In section 2.2, we define the notions of stability and extensibility of a class hierarchy. These notions are based on the ability of subclasses to inherit general methods from their superclasses and customize them for their own use. The customization is performed by writing ad-hoc stand-alone programs as discussed in section 2.3. In section 2.4, we discuss attribute-based access and retrieval of information. In section 3, we discuss the InfoHamess architecture. We begin with an overview (section 3.1), and then concentrate on indexing and searching of InfoHamess repositories (section 3.2) and on browsing information (section 3.3). In section 4 we discuss related efforts. Conclusions and plans for future work are presented in section 5.

2.0 Transparent Data Access through Metadata

An important advantage of InfoHamess is that it provides access to a variety of heterogeneous information without making any assumptions about the location and representation of data. This is achieved by creating metadata and associating it with the original information. Metadata for different media types is often defined as derived properties of the media, which are useful in information access or retrieval [5]. It establishes rela-

relationships between *information units* (defined below) and portions of physical information that present logical units of interest for end-users.

An *information unit* (IU) is a metadata entity that encapsulates a logical unit of information. It represents the lowest level of granularity of information available to InfoHarness. The IU may be associated with a file (e.g., a manpage), a portion of a file (e.g., a C function or a database table), a set of files (e.g., a collection of related bitmaps), or any request for the retrieval of data from an external source (e.g., a database query).

An *InfoHarness object* (IHO) is defined recursively as one of the following:

- A *simple* IHO, composed of a single IU.
- A *collection* IHO that contains a set of references to other InfoHarness objects.
- A *composite* IHO that combines a simple IHO and a set of references to other IHOs.

Each IHO has a unique object identifier that is recognized and maintained by the system. An IHO that contains an IU has to store the locations of both the data and the retrieval method, as well as any parameters needed by the method to retrieve the encapsulated data. In addition, each IHO may contain an arbitrary number of attribute-value pairs (e.g., owner, last update, security information, decryption method).

Each IHO that references a collection of other IHOs stores unique object identifiers of the members of the collection (its *children*). An IHO that both contains an IU and references a collection is called a *composite object*. An **example** of the composite object is this paper's abstract combined with the collection containing references to postscript and latex versions of the full paper.

IHOs that encapsulate indexed collections store information about locations of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not a part of InfoHarness. As a result, an InfoHarness Repository may be created from existing heterogeneous index structures.

An InfoHarness Repository (ThR) is a set of IHOs that are known to a single InfoHarness server. These IHOs are not necessarily members of the same collection. An IHO may be a member of any number of collections (its *parents*). Each IHO that has one or more parents always contains unique object identifiers of its parent objects. An IHO that does not have any parent is unreachable from any other IHO and may only be accessed if it is used as an initial starting point (or *entry point*) in the IHR traversal.

2.1 InfoHarness Class Hierarchy

This section describes the InfoHarness class hierarchy (Figure 1). The *Abstract Classes* provide method sharing between groups of terminal classes. The *Terminal Classes* are classes that cannot be subclassed and are instantiated as InfoHarness objects. Examples of Terminal Classes include encapsulators of external viewers, as well as the Wide-Area Information Service (WAIS) [16] and Latent Semantic Indexing (LSI) [6] indexing tools. The InfoHarness class hierarchy is utilized to instantiate terminal classes to IHOs. We can say that it represents an implicit organization of the metadata.

The abstract class hierarchy (Figure 1) has been constructed to serve two purposes:

1. Support for heterogeneous data and independent indexing technologies.
2. Support for both server-side and client-side run-time processing of information.

The distinction between the *Server Processing* class and the *Client Processing* class is based on the differences in processing information while the rationale for their subclasses is in data representation.

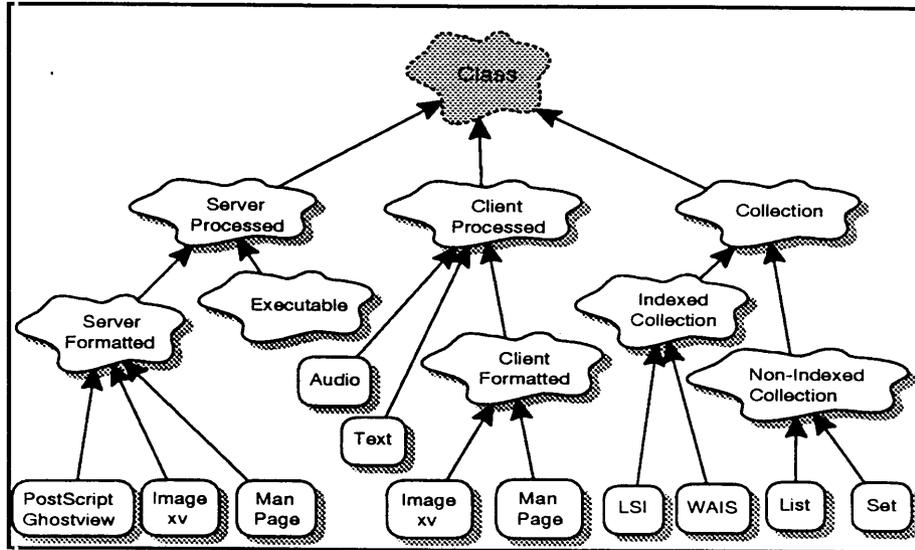


Fig. 1. InfoHarness Class Hierarchy with Sample Terminal Classes.

Server Processing

The *Server Processing* abstract class helps to group IHOs with IUs, which are accessed at mn-time by running a process on the server. The subclasses of this class are the abstract classes *Server Formatted Data* and *Executable*. The subclasses of *Server Formatted Data* serve to access physical data by running external viewers. The abstract class *Executable* represents IUs that encapsulate application programs.

Client Processing

For instances of terminal subclasses of *Client Processing*, data is first transferred to the client and then processed. Most of the data types may be defined by instantiating either a subclass of *Server Processing* or a subclass of *Client Processing*. The exceptions are *Audio* and *Text* that are always defined by instantiating the *Client Processing* class.

- *Audio*: The special treatment of audio files is determined by the need to play the recording on the client machine for it to be heard.
- *Text*: The special treatment of text is for convenience in presenting plain text, as well as documents that use a mark-up language known to the client program.

Collections

Each instance of the *Collection* class stores a set of parent-child relationships between the IHOs. In addition, instances of the *Indexed Collection* class are associated with a physical index that is used at mn-time to select members of the collection. Instances of the *Indexed Collection* class are presented to the end-user through a query interface, **while** instances of *Non-Indexed Collection* are presented as a full list of members.

2.2 Issues of Stability and Extensibility

In this section we discuss the properties of the InfoHarness class hierarchy that are important for supporting new data types and new indexing technologies.

Stability of the Abstract Class Hierarchy

The abstract class hierarchy in InfoHarness is stable in nature; i.e., we do not foresee **the need** for additional abstract classes to model different kinds of processing. Whenever an appropriate terminal class is defined, it inherits data access and representation methods from an existing abstract class. The abstract classes may thus be made part of the server implementation. The terminal class hierarchy is open and extensible, as discussed in the next section.

Extensibility of Data Types

The InfoHarness class hierarchy is open in that new terminal classes may be defined to accommodate the vast variety of information. The new classes utilize methods inherited from the abstract classes. These methods are customized by writing ad-hoc stand-alone programs that are invoked at run-time. In the future, the customization will be performed through interpreting statements of a declarative type definition language.

Terminal classes are not part of the InfoHarness implementation and reflect design choices made by InfoHarness administrators. These choices may be determined either by the availability of data or by the availability of tools used to view the data. It is possible to define two different terminal classes, *Man Page* (descendant of *Server Processing*) and *Man Page* (descendant of *Client Processing*), as subclasses of *Server Formatted Data* and *Client Formatted Data* respectively. These classes may either be used for accessing different sets of man pages or for providing end-users with alternative ways of accessing the same data.

Extensibility of Indexing Strategies

Just as new terminal classes may be defined to support new data types, new terminal classes may also be defined to support independent indexing technologies. The only additional complication is in maintaining the mapping between two different member identifications: the one known to the indexing algorithm and the one known to InfoHarness. It is fairly easy to support WAIS [16], LSI [6,7], and similar indexing technologies by defining appropriate terminal classes. In the present version of InfoHarness, we default to LSI indexing, which is discussed further in section 3.2. In the same section, we discuss the selection of an appropriate indexing strategy depending on the size of a collection, frequency of modification, etc.

2.3 Metadata Extraction

In this section, we discuss the extraction of content-based and content-descriptive metadata. The content-descriptive metadata is composed of IHOs and their relationships. Each IHO in an IHR may have multiple children, as well as multiple parents. The physical data that is associated with information units is not part of the IHR. The generation of an IHR amounts to the generation of both content-descriptive and content-based metadata, the latter by indexing physical information declared to belong to indexed collections.

The IIR generation is supported by the *generator* program, which accepts as inputs both the location and the desired representation of data, and outputs the set of IHOs and their relationships. The *generator* commands may be either written down by InfoHarness administrators, or created automatically by interpreting IRDL statements. Detailed discussion of IRDL is beyond the scope of this paper and may be found in [26].

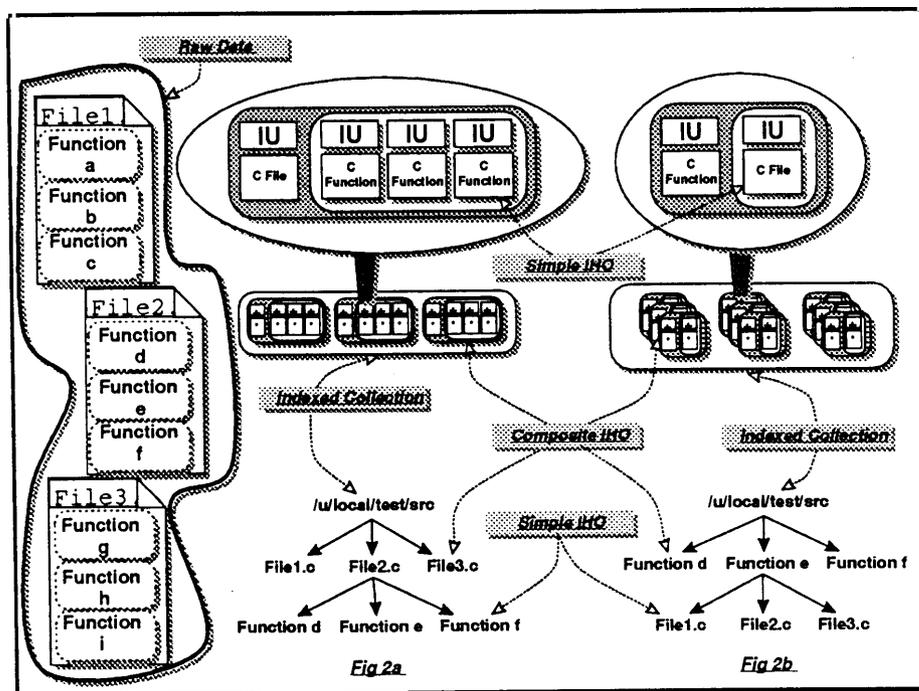


Fig. 2. Metadata Extraction for C code.

Consider the simple example of creating an indexed collection of man pages. Given the location of man pages, their desired mn-time representation, and the desired indexing technology, the generator program performs the following actions:

1. Creates IUs associated with individual man pages.
2. Creates an IHO for every IU created in step 1.
3. Invokes an independent indexing tool to index the man pages.
4. Creates an IHO associated with the index, and adds parent-child relationships from this IHO to each object created in step 2.

By definition, IUs represent the smallest pieces of information retrievable through InfoHarness. Consider the example of an e-mail system that stores messages in individual files. Making this information available through InfoHarness involves defining information units that have one-to-one correspondence with physical files. On the other hand, in the case of *RMAIL* files, a single file contains multiple messages and each message is associated with an information unit.

To better understand the process of metadata extraction, we discuss the metadata generation for C code (Figure 2). To encapsulate individual functions and perform indexing based on comments and function names, the extractor has to perform some basic parsing to recognize comments, function blocks, etc. Function signatures, which uniquely identify individual functions, are stored as values of the *name* attribute of encapsulating objects. Function names are used to qualify file names in the *location* attribute. Similar work has been done in [4] where information about functions and variables is extracted after code analysis. There have also been earlier efforts to index C code based on comments [22].

The next step is to encapsulate and index *C* files, and create parent-child relationships between the collection IHO and the child objects (Figure 2a). The latter are composite objects that reference collections of simple objects, which encapsulate individual functions. Alternatively (Figure 2b), it may be desirable to index the individual functions and establish parent-child relationships between the indexed collection and function IHOs, as well as between function IHOs and file IHOs.

A recent addition to the InfoHarness type library is the *C++* extractor. Since *C++* code has much more information of interest than *C* code, it is correspondingly more complex to write a *C++* extractor. Parse trees are generated and traversed to separate out information units, which encapsulate functions, class definitions, class-subclass relationships, and individual methods.

Additional extractors for unstructured image data and structured data in the form of relational and object oriented databases are being designed. Some of these offer significant research challenges.

2.4 Attribute-based Access and Retrieval

We have considered the following kinds of metadata:

1. content-based metadata,
2. content-descriptive metadata,
 - 2.1. domain-independent metadata,
 - 2.2. domain-dependent metadata.

Examples of content-based metadata include full-text indices of InfoHarness collections. In the case of LSI, these indices have the form of document vectors (section 3.2). The extraction of content-descriptive metadata was discussed in section 2.3. Examples of content-descriptive metadata may include *location*, *ownership*, *creation-date*, etc. The content-descriptive metadata is inferred without performing the semantic analysis.

In the future, we are planning to support more intelligent querying to accommodate naive users. This may require capturing the semantic content of information by describing the meaning and possible usages of individual documents. For this, we need to characterize the *domain* of information represented by a set of IHOs, for example, the domain of software artifacts. While content-based metadata is stored as a set of vectors, the content-descriptive metadata is represented as a set of attribute-value pairs. The attributes may capture domain aspects, as well as characteristics of physical data.

One approach to capturing domain-dependent descriptive metadata for structured databases is described in [18]. The attributes, which are referred to as contextual coordinates, are selected judiciously from domain-specific ontologies. These are then used to construct contexts that capture the assumptions implicit in the design of the object classes in a database. The contexts are used to perform inferences on the information content of the databases without actually retrieving the data.

We are investigating extending the above approach to semi-structured and unstructured data. Additional generalization effort may be well worth it due to the limitations of content-based access, as in the following example:

Consider a paper titled “*Use of Automatically Generated Metadata for Search and Retrieval of Heterogeneous Information*” that is to be presented at the conference in Jyväskylä, and let it be identical to the one that you are reading with the only exception that it does not contain section 2.4. Then, it would not contain the words *conference* and *Jyväskylä*.

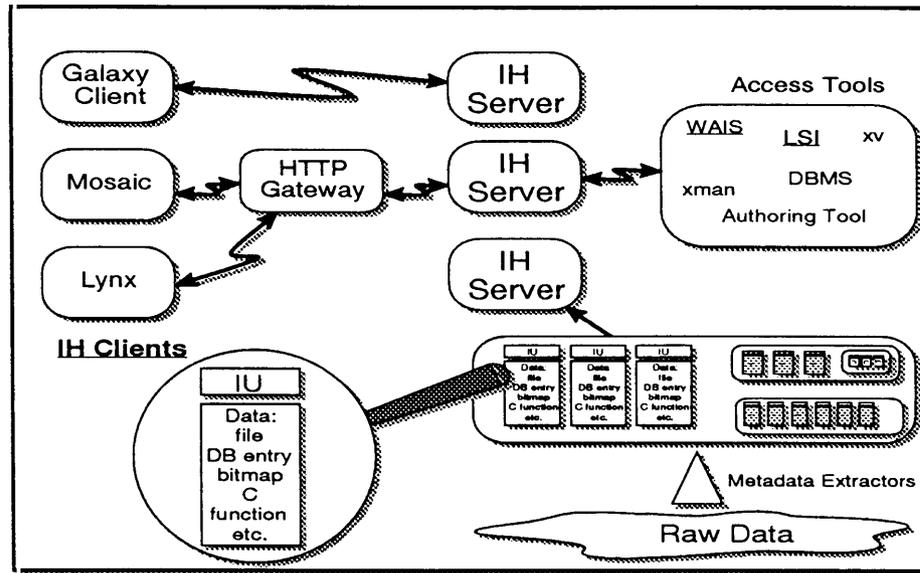


Fig. 3. InfoHarness Architecture.

Consider the following keyword-based query:

Query = *conferenceJyvaskyla*

The above mentioned paper will not be retrieved! The problem may, of course, be remedied by creating additional descriptive attributes and associating them with the paper. Work is also underway to mutually scale results obtained from attribute-based and content-based queries.

3.0 InfoHarness Architecture

The InfoHarness system architecture provides a platform for integrating information in a distributed environment by encapsulating existing and new information, without converting, restructuring or reformatting the physical data. Through this object-oriented encapsulation, the system provides an integrated view and access to diverse and heterogeneous information. The system allows the use of independent tools for accessing, retrieving, browsing, and administering the information encapsulated by IHOs that are stored in InfoHarness repositories.

3.1 Architecture Overview

As shown in Figure 3, the main components of the current implementation of the InfoHarness architecture are:

1. The InfoHarness Server (IH Server) that uses metadata to traverse, search, and retrieve the original information.
2. The HTTP Gateway that is used to pass requests from HTTP clients to the IH Server and the responses back to the clients.
3. The Repository Generator that is used for the off-line generation of metadata that represents the desirable view on the structure and organization of the original information. This metadata is used by the Server for the run-time search and retrieval.

Independent tools for accessing and displaying information (e.g., *xv*, *xman*, etc.), and for indexing information (e.g., Wide-Area Information Server (WAIS), Bellcore's Latent Semantic Indexing (LSI)), are not part of InfoHamess.

At run-time, users of HTTP-compliant clients may issue query, traversal, or retrieval requests that are passed on to the Gateway, which performs the following operations:

1. Parses the request, and reads input information when the request is associated with an HTML form.
2. Establishes a socket connection with the IH Server, generates and sends out a request, and waits for a response.
3. Parses the response, converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser.

The IH Server processes requests based on the meta-information. The same physical data may be treated differently depending on its type. For example, consider two alternative types, both designed to represent man pages: *man_to_html* and *xman*. In the first case, the IH Server will pass the man page location to *aman_to_html* converter and send the generated HTML back to the HTTP gateway. In the second case, the server will pass the man page location directly to the *xman* browser and notify the gateway program.

The Gateway does not perform any format conversions of the original information. It only converts to HTML those parts of the response that either contain the portion of metadata that is currently being searched or traversed by the end-user, or error messages and notifications.

The InfoHamess architecture is open, modular, extensible and scalable. InfoHamess implements the abstract type hierarchy that does not have to be modified to support a new data type, or a new indexing technology. The methods associated with abstract classes are general enough because they are data-driven and may invoke independent programs. The definitions of terminal classes are also data-driven and are not part of the implementation, which makes the system capable of supporting arbitrary information access and management tools (e.g., browsers, indexing technologies, access methods).

The system addresses the goals of scalability and interoperability in a large and geographically distributed environment by supporting multiple servers and multiple clients, and by decentralizing the administration of information repositories.

3.2 Indexing and Searching InfoHamess Repositories

InfoHamess supports both indexed and non-indexed collections. Indices are generated on the bags of words that are contained within portions of physical data that are encapsulated by member IHOs. InfoHamess can support different indexing strategies (Latent Semantic Indexing (LSI), WAIS, etc.). LSI is discussed in the next section, followed by a brief review of alternative indexing technologies and guidelines for their selection.

Latent Semantic Indexing

A fundamental deficiency of many information retrieval methods is that the words used for searching are rarely the same as the words used for indexing. The fact that there are many ways of describing the same object (synonymy) means that many relevant objects will be missed, thus affecting the *recall*. The fact that the same word can be used to refer to different things (polysemy) means that irrelevant objects will be retrieved, thus affecting the *precision*.

LSI's [6,7,28] approach is to take advantage of implicit higher order structure in the

association of terms with documents. It uses the singular-value decomposition that arranges the term space to reflect the major associative patterns in the data and ignore the smaller and less important influences.

Each term and document is represented as a vector in the k-dimensional factor space. A query, just as a document, initially appears as a set of words. It is represented as the weighted sum of its component term vectors and placed at the centroid of its term vectors. The scaled query representation is compared against the representations of all indexed documents, the nearest of which (those with the highest cosines) are selected.

Alternative Indexing Techniques

As mentioned earlier in the paper, InfoHarness can support multiple indexing technologies. We are developing formal criteria and guidelines for selecting the ones that are the most appropriate for particular applications.

LSI is a statistical technique which correlates words on the basis of their occurrence patterns in the documents. It also does the clustering of words based on the statistical analysis, which form the dimensions of the multi-dimensional vector space. The WMS index [16] is a complete inverted index on the document contents and does not involve any statistical analysis. We are currently investigating the following criteria for choosing between these technologies:

- *Size of a Collection:* The WMS indexing technique is preferred if there are just a few documents in the collection. Statistical techniques like LSI require a large sample space to prevent meaningless answers at the time of query processing.
- *Frequency of Modification:* WMS is also preferred if the collection is modified frequently (addition, deletion and update of documents). This is because extensive statistical computations are required by LSI to recompute the word clusters.
- *Types of Queries:* The WMS index is suitable for keyword-based queries but not for finding documents that do not contain the exact keywords (e.g. searching for *car* it may not be possible to retrieve documents containing the word *automobile*). The LSI index supports such “semantic” searches.
- *Domain Structure of Information:* It would be advantageous to use LSI if all documents are from the same domain or display a typical usage correlation pattern or structure. If the documents are chosen at random from different domains, WAIS is likely to work better.

3.3 Browsing Information

This section addresses browsing physical data, which may range from plain text to multimedia information. We distinguish between browsing metadata and browsing the encapsulated physical data. Browsing methods are determined by types, which are assigned by the metadata extractors at the pre-processing stage.

Each IHO contains relevant information about the encapsulated physical data, as well as the relationships with other IHOs. This structure is presented to the user, who can choose to retrieve the physical data or to traverse parent-child relationships. Browsing methods act upon the data retrieval request after determining the physical location of information. A browsing method may or may not invoke a presentation tool. Independent browsers are used to browse through postscript documents, image files, etc.

The relationships between information units and physical files are different for types, as described in section 2.3. The extractor methods understand these relationships and generate the metadata accordingly. The search and browsing methods obtain this

very same knowledge from IHOs and their relationships. The browsing methods are specific to the representation of the physical information. For example, an image viewer may be invoked for image files and an audio player for sound files. The user need not have any knowledge about ways of launching the appropriate tool. Hypertext browsing can also be supported if the original document is a hypertext.

3.4 Query Processing in InfoHarness

In the previous sections we have identified two ways of accessing IHOs in InfoHarness repositories. They are as follows:

- *Attribute-based Access*: This has been discussed in section 2.4 of this paper and involves querying documents based on their attributes, which may contain information not captured in their contents (e.g., ownership, publisher, etc.).
- *Content-based Access*: This involves querying for documents based on their contents (e.g., presence of keywords, presence of patterns, etc.). Querying indices built from the contents of documents is an example of content-based access. We are planning to further simplify adding support for third party indexing technologies.

We are currently investigating different ways of meaningfully combining results obtained from attribute-based and content-based queries.

4.0 Related Work

Hypermedia systems like Intermedia [30] focus more on browsing than on retrieval. As for database systems, the original data has to be converted into internal representations that are usually proprietary. On the other hand, traditional file systems lack the ability to represent inter-file relationships. Huge and growing amounts of heterogeneous information call for a more versatile system.

InfoHarness shares some of its objectives with RUFUS [261]. The RUFUS system has an extensible object-oriented data model, storage system, and associated search and display methods for a variety of predefined file types. RUFUS users can also search, browse, filter and display the imported data objects. The system automatically classifies data files and extracts type-specific attributes. In InfoHarness, its information units may be associated with files, sets of files, or portions of these files. This approach is more flexible in providing finer control at interpreting data.

Metadata is being increasingly used by researchers in multimedia, text and structured databases as an aid in the quest for seamless interoperability. Chen et al [5] define metadata as derived properties of the media which are useful for information access or retrieval. In the meta-database project at RPI [12], a metadata management approach is adopted to achieve a global synergy between various component databases. Bohm and Rakow [3] have classified metadata according to their nature and related it with their different intended purposes. The comparison of this classification with the approach used in InfoHarness is discussed further in the section. Kiyoki et al [19] implement a semantic associative search for images based on the keyword metadata representing the user's impression and the image's content. Anderson and Stonebraker [1] have developed a metadata schema for satellite images. Jan and Hampapur [13] have proposed an intermediate representation for audio-visual information.

In InfoHarness, we have emphasized the extensive use of automatically generated metadata. Bohm and Rakow [3] have proposed a novel classification of metadata and have drawn a distinction between the metadata and its organization. They have also es-

tablished the relation between the intended use of metadata and its classification. The same perspective is reflected in the hierarchical organization of InfoHamesh objects and classes as illustrated in Figure 1. The overlap between the classification in [3] and our approach is as follows:

- *Representation of Information Types:* This is achieved in InfoHamesh through various encapsulators for different kinds of information. It is also one of the rationales for the proposed organization of metadata. In [1], metadata is used to support two perspectives of satellite image data: a computer scientist's and an earth scientist's.
- *Content-based Metadata:* This type of metadata refers to the metadata extracted from the physical information based on its content. Vectors associated with textual documents in an LSI-indexed collection are examples of content-based metadata. Vectors proposed for text and audio documents in [10], time-indices of keywords and location of keywords in a text image based on word spotting [5], and the data features in [13] also follow the content-based approach.
- *Content-Descriptive Metadata:* This is being planned as an extension to InfoHarness (section 2.4). Content-descriptive metadata has been used in [18] for information resource discovery and query processing in Multidatabases. The domain-dependent Q-features and the content-independent meta-features in [13] are examples of content-descriptive metadata for audio-visual information.
- *Document Composition:* This type of metadata is created at the time of extraction when the parent child relationships between different IHOs are being established. This perspective is reflected in the structure of composite IHOs.
- *Document Location:* This information is represented as an attribute for all kinds of IHOs supported by the InfoHamesh system.
- *Document Collection Metadata:* The information associated with a full-text index is a type of metadata associated with the collection of documents as a whole.
- *Presentation Requirements:* Different technical settings on the client and the server sites demand more flexible document-processing mechanisms. In the InfoHamesh system, the idiosyncracies are captured as metadata. This perspective is reflected in the abstract classes Client Processing and Server Processing in Figure 1.
- *Browsing:* In the InfoHamesh system, browsing takes place through the association relationships between information units and physical data. These associations are captured by the metadata (section 3.3). In [11], the metadata is represented through objects with their associated relationships and attributes. These relationships are used for browsing through the associated media objects (images and video).

5.0 Conclusions and Future Work

The important advantage of InfoHamesh is in providing flexible access to arbitrary heterogeneous information without any relocation or reformatting. The InfoHamesh Abstract Type Hierarchy is stable in the sense that it does not have to be modified to support new user-specific terminal classes. This hierarchy has been constructed to achieve the dual flexibility in the representation, as well as the presentation of data.

Our current work is directed toward further automation of the generation of InfoHamesh repositories. However, it is not our intention to direct the generation of the repositories by the analysis of physical data alone. We intend to develop a declarative Repository Definition Language [26]. Statements of this language, combined with the physical data, together would determine the structure of InfoHamesh repositories.

We are also performing investigations aimed at improving the scalability of search by meaningfully combining the results of querying independent (possibly heterogeneous) indices that reference different collections of objects.

6.0 Acknowledgements

The authors want to thank Chuniki Basu, Satish Thatte, Gomer Thomas, and Andrew Werth for their comments to the draft of the paper.

References

- 1 J. Anderson and M. Stonebraker, "SEQUOIA 2000 Metadata schema for Satellite Images", (to appear) SIGMOD Rec., special issue on Metadata for Digital Media, December 1994.
- 2 T. Berners-Lee et al, "World Wide Web: The Information Universe", Electronic Networking: Research, Applications and Policy", 1(2), 1992.
- 3 K. Bohm and T. Rakow, "Metadata for Multimedia Documents", (to appear) SIGMOD Record, special issue on Metadata for Digital Media, December 1994.
- 4 Y-F. Chen, M. Nishimoto and C. Ramamoorthy, "The C information abstraction system", IEEE Transactions on Software Engineering, March 1990.
- 5 F. Chen, M. Hearst, J. Kupiec, J. Pederson and L. Wilcox, "Metadata for Mixed-Media Access", (to appear) SIGMOD Record, special issue on Metadata for Digital Media, December 1994.
- 6 S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer and R. Hashman, "Indexing by Latent Semantic Indexing", Journal of the American Society for Information Science, 41(6), 1990.
- 7 S. T. Dumais, G. W. Fumas, T. K. Landauer, S. Deerwester, and K. Harshman, "Using latent semantic analysis to improve access to textual information", Proceedings of the 1988 Cifi Conference, 1988.
- 8 G. Fischer and C. Stevens, "Information access in complex, poorly structured information spaces", Proceedings of the 1991 Cell Conference, 1991.
- 9 F. Garzotto, P. Paolii, and D. Schwabe. "HDM - A Model-Based Approach to Hypertext Application Design", ACM Trans. on Inform. Systems, 11(1), 1993.
- 10 U. Glavitsch, P. Schauble and M. Wechsler, "Metadata for Integrating Speech Documents in a Text Retrieval System", (to appear) SIGMOD Record, special issue on Metadata for Digital Media, December 1994.
- 11 W. Grosky, F. Fotouhi and I. Sethi, "Content-Based Hypermedia - Intelligent Browsing of Structured Media Objects", (to appear) SIGMOD Record, special issue on Metadata for Digital Media, December 1994.
- 12 C. Hsu, "The Meta-database Project at Renesselaer", SIGMOD Record, special issue on Semantic Issues in Multidatabases, 20(4), December 1991.
- 13 R. Thin and A. Hampapur, "Representations for Video Databases", (to appear) SIGMOD Record, special issue on Metadata for Digital Media, December 1994.
- 14 Y. Kane-Esrig, L. A. Streeter, W. Keese, and G. Casella, "The relevance density method in information retrieval", Proceedings of the 4th International Conference on Computing and Information, 1992.

- 15 Y. Kane-Esrig, L. Shklar, and C. St. Charles, Using Multiple Sources of Information to Search a Repository of Software Lifecycle Artifacts, Proceedings of the Bellcore Conference on Electronic Document Delivery, NJ, May 1994.
- 16 B. Kahle and A. Medlar, "An Information System for Corporate Users: Wide Area Information Servers", Connexions - The Interoperability Report, 5(11), Nov. 1991.
- 17 V. Kashyap and A. Sheth, "Semantics-based Information Brokering: A step towards realizing the Infocosm", Technical Report DCS-TR-307, Department of Computer Science, Rutgers University, March 1994.
- 18 V. Kashyap and A. Sheth, "Semantics-based Information Brokering", Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM), Gaithersburg, MD, November 1994.
- 19 Y. Kiyoki, T. Kitagawa and T. Hayama, "A meta-database System for Semantic Image Search by a Mathematical Model of Meaning", (to appear) SIGMOD Record, special issue on Metadata for Digital Media, December 1994.
- 20 T. Landauer, D. Egan, M. Lesk, C. Lochbaum and D. Ketchum, "Enhancing the usability of text through computer delivery and formative evaluation: the Super-Book Project", In C. McKnight, A. Dillon and J. Richardson, (eds) "Hypertext: A Psychological Perspective", Chichester: Ellis Horwood, 1993.
- 21 C.J. Matheus, P.K. Chan, and G. Piatetsky-Shapiro, "Systems for Knowledge Discovery in Databases", IEEE Trans. on Knowledge and Data Eng., Dec. 1993.
- 22 V. Sembugamoorthy, L. Streeter, and Mary Leland, "Igrep: A real World Prospective on Locating Software Artifacts for Reuse", Proceedings of the 5th Annual Workshop on Software Reuse (WISR'92), Palo Alto, CA, 1992.
- 23 A. Sheth and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", ACM Comp. Surveys, 22(3), 1990.
- 24 L. Shklar, "XReuse: Representation and Retrieval of Heterogeneous Multimedia Objects", Proc. of Bellcore Object-Oriented Symposium, Arlington, VA, June 1993.
- 25 L. Shklar, S. Thatte, H. Marcus, and A. Sheth, "The InfoHarness Information Integration Platform", Advance Proceedings of the Second International WWW Conference '94, Chicago, October 17-20, pp. 809-819, <http://www.ncsa.uiuc.edu/SDGI/IT94/Proceedings/Searching/shklar/shklar.html>.
- 26 L. Shklar, K. Shah, and C. Basu, "Putting Legacy Data on the Web: A Repository Definition Language", To appear in the Proceedings of the Third International WWW Conference '95, April 1995, Darmstadt, Germany, <http://www.igd.jhg.delwww/www95/www95.html>.
- 27 K. Shoens, A. Luniewski, P. Shwartz, J. Stamos, and J. Thomas, "The Rufus System Information Organization for Semi-Structured Data", Proceedings of the 19th VLDB Conference, Dublin, Ireland, 1993.
- 28 L. A. Streeter and K. E. Lochbaum, "Who knows: a system based on automatic representation of semantic structure", Proceedings of RIAO 88: User-oriented context-based text and image handling, M.I.T., Cambridge, MA, 1988, pp. 379-388.
- 29 John R. Rymer, "Distributed Object Computing", Distributed Computing Monitor, Vol. 8, No. 8, Boston, 1993.
- 30 N. Yankelovich, B. Haan, N. Meyrowitz and S. Drucker, "Intermedia: The concept and construction of a seamless information environment", IEEE Computer, 21(1), January 1988.