Using Metadata to Query Passive Data Sources

Patrick Martin, Wendy Powley and Andrew Weston
Department of Computing and Information Science, Queen's University
Kingston, Ontario, Canada K7L 3N6
{martin, wendy, weston}@qucis.queensu.ca

Abstract[*]

In the not too distant past, the amount of online data available to general users was relatively small. Most of the online data was maintained in organizations' database management systems and accessible only through the interfaces provided by those systems. The popularity of the Internet, in particular, has meant that there is now an abundance of online data available to users in the form of Web pages and files. This data, however, is maintained in passive data sources, that is sources which do not provide facilities to search or query their data. The data must be queried and examined using applications such as browsers and search engines. In this paper we explore an approach to querying passive data sources based on the extraction, and subsequent exploitation, of metadata from the data sources. We describe two situations in which this approach has been used, evaluate the approach and draw some general conclusions.

1. Introduction

Prior to the recent "information explosion" brought on by the Internet, the amount of online data available to general users was relatively small and was composed largely of data stored in organizations' database systems. We will refer to these database systems as *active data sources* since their data is accessible only through the interfaces provided by those systems.

The growth of the Internet has meant that there is now an abundance of online data available to users in a variety of forms such as Web pages, software, sounds, images, postscript files, library catalogs, user directories, and scientific data. This data, however, is maintained in what we will call *passive data sources*, that is sources which do not provide facilities to search or query their data. The data must be retrieved using "third-party" applications such as browsers and search engines. A key problem currently being addressed by many researchers is how to make effective use of the wealth of data available from these passive data sources. For example, particular topics of interest with respect to the World Wide Web (or simply the Web) include resource discovery [4,20], data mining [7] and query tools [10,15].

One obstacle to making effective use of the data in passive data sources is the lack of facilities to query the data. Passive data sources do not provide languages, tools or Application Programmer Interfaces (APIs) to support sophisticated querying. A second obstacle is the data itself. Data in passive sources has the following characteristics [4]:

- it is *unstructured*, that is, there is no schema, or machine-readable description of the structure, available from the data source;
- it is *heterogeneous*, both across data sources and within single sources;
- it is *inconsistent* because most data contained in passive data sources is dynamic, for example Web pages;
- it is *incomplete* because additional properties of an object can often be obtained by combining data from several sources.

In this paper we outline an approach to querying passive data sources which is based on the extraction, and subsequent exploitation, of metadata from the data sources. We define *metadata* to be descriptions of the properties of, and the relationships present in, the data and the data sources. The approach has been used successfully in two different situations. Other examples of approaches using metadata to query passive data sources only consider a particular collection of passive data sources, usually the Web. We examine the characteristics of our two examples and propose a general framework which can be used with any collection of passive data sources.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 presents our metadata model and repository. Section 4 describes two examples where our approach has been used. The first example is a repository system for configuration

[*] This work is supported by IBM Canada Ltd. and the National Science and Engineering Research Council of Canada.

management of distributed applications and the second example is a system for querying the World Wide Web. Section 5 contains a discussion of our approach and indicates how the approach used in the two examples can be generalized. Section 6 presents our conclusions.

2. Related Work

As we stated earlier, the problem of effectively exploiting the data available in passive data sources is currently receiving a great deal of attention. The focus of most of the attention is the collection of data sources on the World Wide Web. There are currently a number of well-known search engines available for the Web including Lycos, AltaVista, InfoSeek, OpenText and WebCrawler [11]. Most of these search engines are keyword-based and operate by retrieving documents from the Web, indexing them, and then storing them in a local database.

There are a number of problems with popular Web search engines which are being addressed by different research efforts. For example, the Harvest system [5] is addressing the problems of reducing the load search engines place on Web servers and of duplication of effort by current indexing systems. There are also several projects, including WISE (World Wide Web Index and Search Engine) [20], W3QS [10], WebSQL [15], the Information Manifold [12] and our own work [19], which overcome limitations in current search engines with more powerful query languages which exploit Web document structure as well as contents.

In addition to Web-based applications, metadata is used in general collections of unstructured data where semantic properties cannot be directly inferred from the data. For example, database systems maintaining multimedia data such as audio or video rely on metadata to describe the digital data [9]. Metadata is also often used to support integration of heterogeneous data sources. For example, metadata is used in multidatabase systems [2,14] and information resource management systems [8] to describe the data present in the component databases and the relationships that exist among that data.

3. Metadata Model and Repository

Our approach to querying passive data sources is based on the extraction, and subsequent exploitation, of metadata from the data sources. We chose this approach because it can apply to a variety of types of data sources. We assume that the data in the data sources, while not easily queried by users, can be processed by customized extraction tools. If the data is in a digital form that can not be analyzed then the metadata must be provided manually.

The metadata includes descriptions of the properties of, and the relationships present in, the data. The metadata is organized according to a *metadata schema*, which is defined using a high-level *metadata model*. An instance of a metaschema is called a *metadatabase* which is stored in a repository. Once the schema is defined, tools, which are based on the schema, are built to extract the metadata from the data sources, store it in a metadatabase, and then query the repository to facilitate access to the data sources. By querying the metadatabase generated for a collection of passive data sources rather than data sources themselves, users are able to conveniently formulate queries which specify conditions on the structure and the semantics of the data as well as its content. This allows users to arrive at their desired results in a more direct and effective manner.

3.1. Metadata Model

The main role of the metadata model is to represent the different types of metadata concepts, and their relationships, present in the particular application. We use a structurally object-oriented model, based on the Telos language[16], to define the structures of our metadata model. We chose an object-oriented model for two main reasons:

1. the entities to be modeled may have complex structures and relationships;
2. there will be a large number of similar entities which may be grouped into classes.

We chose to base our model on Telos instead of an object-oriented database system because Telos, as a conceptual modeling language, is particularly convenient for expressing complex relationships among the data. It is also a high-level language and so is independent of any programming language.

The metamodel uses the following constructs:

- *attribute*: An attribute is a particular property of an object. Each attribute has a *type* which may be a *primitive type*, such as string, integer or real, or a user-defined class. In the latter case, the value for the attribute is a *reference* to an instance of that class. Attributes may be *single-valued* or *multi-valued*.
- *object*: An object is an identifiable collection of attribute values which represents a metadata concept. Every object has a unique object identifier or name.
- *class*: A class is a collection of objects that share common properties. An object is related to a

particular class via the *instance-of* relationship. Classes are related via the *is-a* relationship.

- *metaclass*: A metaclass is a collection of classes that share common *categories* of attributes where a category groups attributes related to a particular aspect of an object.

3.2. Metadata Repository

The *Metadata Repository* (MDR) is shown in Figure 1. It is a basic client-server architecture. The *MDR Server* has two components: the Telos Repository, which provides the back-end database, and the MDR Server Interface, which takes requests from MDR Clients and translates them into requests to the Telos Repository. The *MDR Client* also has two components: an application that requires access to the MDR and the MDR Client Library, which is an API to the MDR.

The Telos Repository used for the MDR prototype is the University of Toronto implementation of the Telos language [16]. This implementation uses ObjectStore[1] as the underlying storage mechanism. The MDR Server Interface, which provides functions for connecting to the repository, submitting and retrieving objects, and querying the MDR, is implemented in C++ and runs on an IBM RS/6000[2] computer on top of OSF DCE [18]. The back-end of the interface communicates with the Telos Repository via the Telos message Bus (TMB) API. Requests and results are passed along the TMB in the form of *s-expressions* which are strings that are parsed and understood by the Telos Repository.

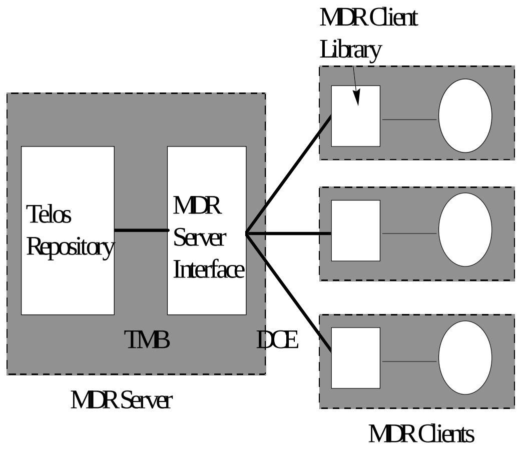


Figure 1: Metadata Repository

Although theoretically MDR Clients could use the TMB API directly to access the Telos Repository, we decided to build an intermediate layer (the MDR Server Interface) between the Telos Repository and the MDR Clients for the following reasons:

- **Database Independence -** The MDR Server Interface buffers the MDR clients from the specifics of Telos. This will allow us to change the underlying storage mechanism with minimal modifications to the overall system. Only the back-end of the MDR Server Interface would need to be modified in order to accommodate such a change. The MDR Client Interface would remain stable, thus requiring no change to the possibly numerous MDR Clients.

- **Multiplexing -** The TMB allows only a single client to access the Telos Repository at a time. The MDR Server uses threads to service and coordinate multiple clients, sending one request at a time to the Telos Repository.

- **Extended Query Capabilities -** The Telos Repository provides very limited query capabilities. The MDR Server Interface extends these capabilities to include conjunctive queries based on *instance-of* conditions, *is-a* relations and queries by attribute value. The MDR Server Interface generates a query which can be processed by the Telos Repository, sends the request to the repository, then filters the result to return only the objects requested by the MDR Client.

Clients, which consist of the Client Library and a MDR application, communicate with the MDR Server via DCE RPCs. The Client Library is implemented in C++ and uses the C++ API provided by the MDR Server Interface. It presents applications with a view of the MDR which is consistent with metadata model and provides functions to create a new metadatabase, connect to an existing metadatabase, disconnect from an active

- [1] ObjectStore is a trademark of Object Design, Inc.

- [2] RS/6000 is a trademark of International Business Machines Corp.

metadatabase, modify the contents of the active metadatabase, and query the active metadatabase.

4. Examples

We consider two example applications of our approach. For each example we outline the particular problem that was addressed, we provide a brief description of the metadata schema developed for the problem, and we discuss the tools built based on the metadata schema.

4.1. Configuration Management of Distributed Applications

The Problem

The management of large complex systems involves the maintenance of a large number of components and requires an understanding of the properties of these components and of the relationships between them. Configuration management, in particular, is responsible for the collection and maintenance of descriptive and location information about system components and system structure.

The configuration information of a distributed application is divided into the “runtime” information and the “code-view” information [13]. The runtime information is collected dynamically and describes the state of an execution of the application. The code-view information describes the static organization and construction of the software that makes up the application. This configuration information is metadata that describes the various types of data (code files, data files, execution states, etc.) associated with a distributed application.

The code-view information, however, is not easily obtained since it is embedded in the various files that make up the application and in the directory structure holding the files. Currently, users have to rely largely on operating system facilities to acquire the information. We applied our metadata approach to the problem of acquiring code-view information from the application files. We consider the files as the passive data sources and the code-view information as the metadata. Based on this assumption we built tools to extract configuration information from the application files and then to browse and query the metadata which has been stored in the MDR. These tools are in turn used by a systems administrator to help manage the distributed application [3].

The Metadata Schema

A portion of the metadata schema for this example is shown in Figure 2. A complete description of the schema is given in [13]. The nodes above the horizontal line are metaclasses and the nodes below it are simple classes. Simple classes are *instances-of* metaclasses. Classes at the same level are related by the *is-a*, or specialization, relationship. The metaclasses define the categories of attributes present in the different groups of simple classes.

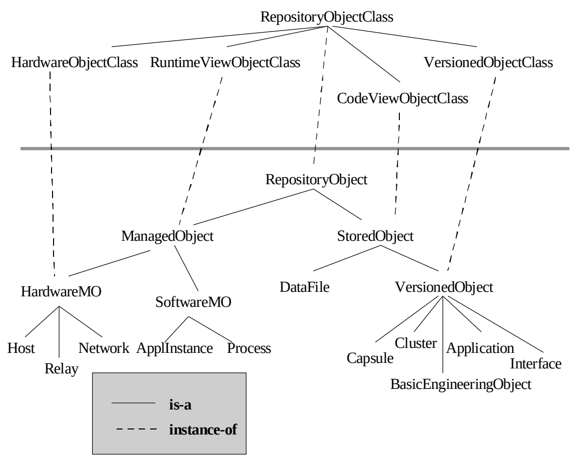


Figure 2: Configuration Management Metadata Schema

The root metaclass **RepositoryObjectClass** defines the categories of attributes found in all classes. The metaclasses under **RepositoryObjectClass** introduce attribute categories appropriate to the different collections of objects represented by the simple classes, that is hardware objects (**HardwareObjectClass**) such as nodes and networks, runtime objects (**RuntimeObjectClass**) such as processes and application instances, and code-view objects (**CodeViewObjectClass**) such as source files and executables. The **VersionedObjectClass** metaclass defines the categories of attributes related to the version information that is required for some of the code-view objects.

The code-view configuration information is represented in the metadata schema mainly by a subtree of classes with the class **StoredObject** at the root of the subtree. **StoredObject** defines the set of attributes related to the physical file that stores the object. The different kinds of files associated with an application include source files (**BasicEngineeringObject**), object files (**Cluster**), executables (**Capsule**), interface definition files (**Interface**), data files (**DataFile**) and makefiles and startup scripts (**Application**).

The Tools

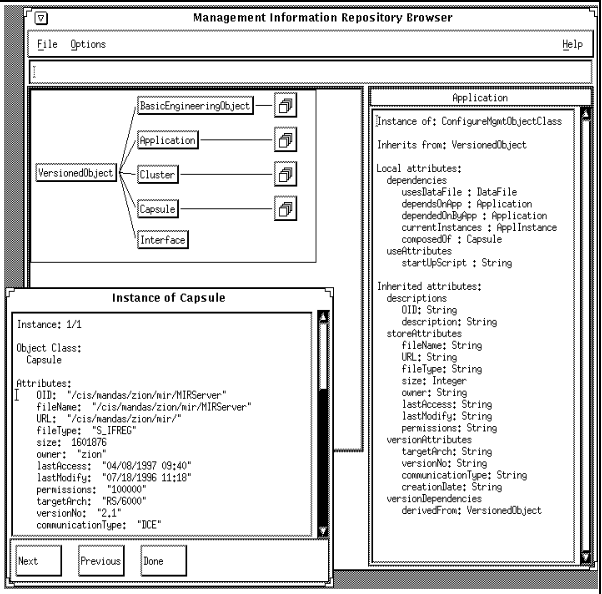
Several configuration management tools which use metadata stored in the MDR have been built. Here we describe the *MDR Browser* which allows a user to examine the contents of the MDR and the *Code View Data Extractor (CVDE)* which automatically populates the MDR with metadata describing the code-view of a distributed application.

The metadata associated with the code-view of a distributed application is maintained in a collection of passive data sources, namely the code files associated with the distributed application which may include many source files, executables and interfaces. Therefore, the metadata to describe the code-view of a distributed application is likely to be extensive. To manually populate the MDR with the metadata would be tedious. The CVDE automates the extraction of the code-view metadata from the various files comprising a distributed application.

The CDVE prompts the user for general information about the distributed application such as the application name, the top-level directory of the application code files, the type of middleware used (we currently handle DCE [18] and Corba [17] applications) and information about each executable associated with the distributed application. It uses this information to extract the metadata from the application files and generates instances of **Application**, **Cluster**, **Capsule**, **BasicEngineeringObject**, and **Interface** objects.

The MDR Browser presents a graphical view of the contents of the MDR allowing users to browse metadata schema and query the metadata stored in the MDR. Figure 3 shows the MDR browser interface. Users can employ the graphical display to browse the metaclass/class hierarchy. Selected class or metaclass descriptions are shown in the text window to the right. The class description shows the attributes and their types as well as the *instance-of* and *is-a* relationships of the class.

Classes which have associated data objects (that is, instances of the class) are indicated by the instances icon (as seen to the right of the **BasicEngineeringObject** object in Figure 3). Selecting the icon produces a text box containing the information for all the data objects associated with the class. In Figure 3, the **Capsule** instances icon has been selected and one of the data objects is shown. The user may cycle through the list of



```
Instance of: ConfigureMgmtObjectClass

Inherits from: VersionedObject

Local attributes:
  dependencies
     usesDataFile : DataFile
     dependsOnApp : Application
     dependedOnByApp : Application
     currentInstances : ApplInstance
     composedOf : Capsule
  useAttributes
     startUpScript : String

Inherited attributes:
  descriptions
     OID: String
     description: String
  storeAttributes
     fileName: String
     URL: String
     fileType: String
     size: Integer
     owner: String
     lastAccess: String
     lastModify: String
     permissions: String
  versionAttributes
     targetArch: String
     versionNo: String
     communicationType: String
     creationDate: String
  versionDependencies
     derivedFrom: VersionedObject
```

```
Instance: 1/1

Object Class:
  Capsule

Attributes:
    OID:  "/cis/mandas/zion/mir/MIRServer"
    fileName:  "/cis/mandas/zion/mir/MIRServer"
    URL:  "/cis/mandas/zion/mir/"
    fileType:  "S_IFREG"
    size:  1601876
    owner:  "zion"
    lastAccess:  "04/08/1997 09:40"
    lastModify:  "07/18/1996 11:18"
    permissions:  "100000"
    targetArch:  "RS/6000"
    versionNo:  "2.1"
    communicationType:  "DCE"
```

Figure 3: MDR Browser

data objects, one at a time, by using the "Next" and "Previous" buttons.

In addition to browsing the MDR, the browser provides facilities to query the contents of the MDR via a graphical user interface. The user may formulate conjunctive queries based on instance-of conditions, is-a relations and queries based on attribute value.

4.2. Querying the World Wide Web

The Problem

As we discussed above, the Web is the largest example of a collection of passive data sources. The existing search engines for the Web are mostly keyword-based and operate by retrieving documents from the Web, indexing them, and then storing them in a local database. This local database is then used to answer user queries.

There are a number of shortcomings with existing search engines. The work described in this example, which is described in detail in [19], addresses shortcomings in the query capabilities of the search engines. Existing search engines use either keyword or full-text indexing techniques. Neither of these techniques capture the internal structure of the Web documents, defined by their HyperText Markup Language (HTML) tags, or the external structure of the portion of the Web containing the documents which is defined by the hypertext links connecting the documents. We have used our metadata approach to provide a tool which handles queries involving both the content and the structure of the Web documents.

The Metadata Schema

The metadata schema for this example is shown in Figure 4. **WebObjectClass** defines the category **descriptions** which can be found in all simple classes since **WebObjectClass** is the root of the class hierarchy. The metaclass **AuthorObjectClass** defines the categories of attributes used to describe the authors of Web documents and the metaclass **ComponentObjectClass** groups together the different classes of objects used as components of the primary metadata objects - documents and hypertext objects.

The different types of documents are grouped under the metaclass **DocumentObjectClass**. The types of documents handled by the prototype system are postscript (class **PSFile**), text (class **TextFile**) and HTML (class **HyperTextFile**). These three classes of documents are all a specialization of the general class **Document** which defines the attributes common to all documents, such as URL, date, author and subject.

The classes of hypertext objects are grouped under the metaclass **HyperTextObjectClass** and are used to represent the metadata describing the structure of the Web documents. The objects that we consider are headings (class **Heading**), which define the structure of an HTML document by encompassing text with the tags *<H1>...</H1>* to *<H6>...</H6>;* hyperlinks (class **HyperLink**) which are denoted by the *href* tag, and multimedia objects (class **GraphicLink**) which include in-line images, such as GIF files, and video or audio files. The third subclass of the **UrlLink** class, **DocumentLink**, is used for hyperlinks that are not **HyperLink** or **GraphicLink** objects.

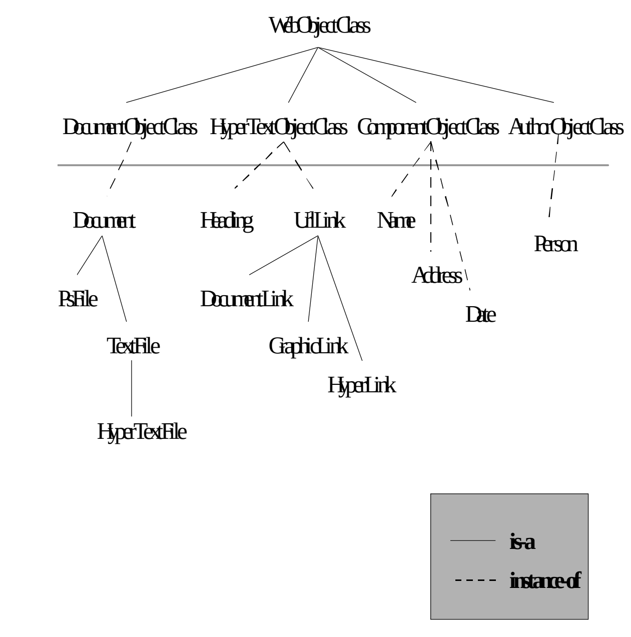


Figure 4: Web Document Metadata Schema

The Tools

The structure of the Web query system is shown in Figure 5. The metadata collection tool is a robot which is a combined Web crawler and document indexer. The robot's execution is broken down into units called *runs*. For each run, the robot is given an initial URL and the total number of documents to be processed in the run. The robot starts a run by retrieving the document at the initial URL. As a document is processed the URLs of all hyperlinks are stored and, once all metadata for that document is collected, the hyperlinks are processed in a breadth-first manner.

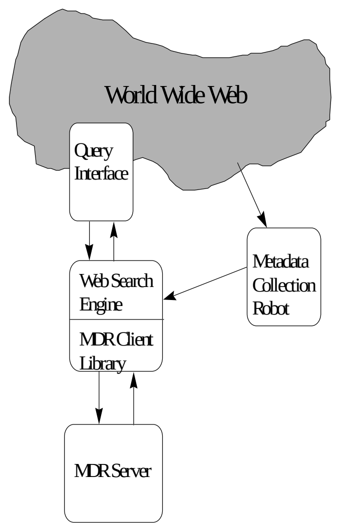
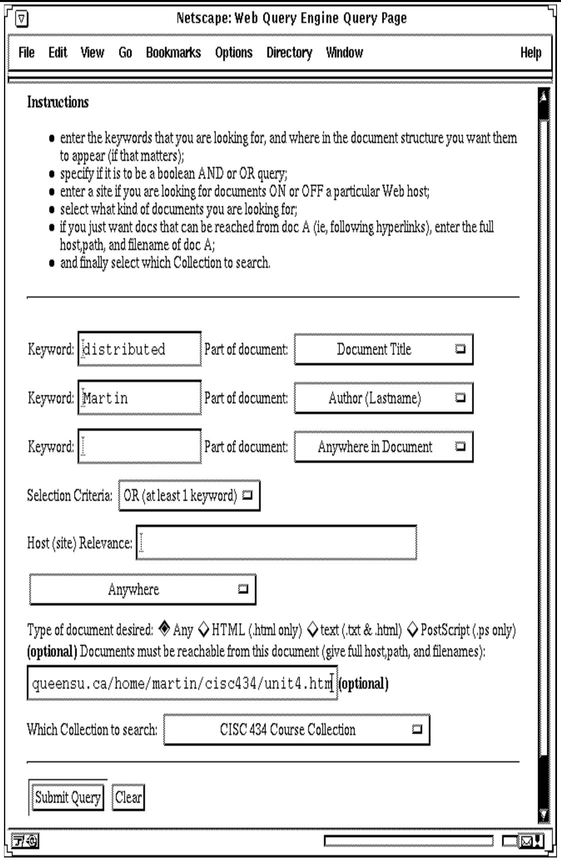


Figure 5: Web Query System Structure

When a document is downloaded from the Web the robot parses the documents based on the HTML tags that identify data corresponding to the classes in the metadata schema. The data associated with the tags is transformed into attribute values for a new metadata object. When the object's description is complete, the robot uses functions provided by the MDR Client Library to insert the object into a metadatabase.

The metadata query tool is composed of a Web-based query interface and a query engine which interacts with the MDR. The interface component consists of an HTML form and a cgi-bin script. An example of the HTML interface is shown in Figure 6. The user can specify conditions on the structure of a document or on relationships with other documents or locations such as keywords in particular parts of the document, locations for the document and type of document. The user then submits the query to the query engine.


Netscape: Web Query Engine Query Page
File Edit View Go Bookmarks Options Directory Window Help

Instructions

- enter the keywords that you are looking for, and where in the document structure you want them to appear (if that matters);
- specify if it is to be a boolean AND or OR query;
- enter a site if you are looking for documents ON or OFF a particular Web host;
- select what kind of documents you are looking for;
- if you just want docs that can be reached from doc A (ie, following hyperlinks), enter the full host,path, and filename of doc A;
- and finally select which Collection to search.

Keyword: distributed Part of document: Document Title

Keyword: Martin Part of document: Author (Lastname)

Keyword: Part of document: Anywhere in Document

Selection Criteria: OR (at least 1 keyword)

Host (site) Relevance:

Anywhere

Type of document desired: [✓ Any] HTML (.html only) text (.txt & .html) PostScript (.ps only)

(optional) Documents must be reachable from this document (give full host,path, and filenames):

queensu.ca/home/martin/cisc434/unit4.htm **(optional)**

Which Collection to search: CISC 434 Course Collection

Figure 6: Web Query Interface

The query engine transforms the information from the query interface into a query to the appropriate metadatabase in the MDR. The query conditions map to conditions on metadata objects in the metadatabase. References to metadata objects that match the conditions are collected from the MDR and references to the corresponding documents passed back to the user in the form of an HTML document. At this point the user can chose to view particular documents or to refine the query.

5. Discussion

The two examples demonstrate the usefulness of our metadata-based approach to querying passive data sources. In both cases we were able to successfully build tools which extracted metadata from the collection of passive data sources and then used that metadata to enhance the users' abilities to query the data sources.

The approach is, we believe, applicable to any type of passive data source. Text data sources are the best candidates since it is more likely that we can build automated, or at least semi-automated, metadata extraction tools for text data. The approach will work for general digital data but the metadata may have to be manually constructed. The approach is also useful for situations in which data about an object may be in multiple data sources. For example, configuration management about a Capsule object, that is an executable file, may be found in multiple files. Our approach is able to recognize this fact and to allow the integration of the metadata from the multiple sources into a single metadata object. The metadata makes the relationship between the multiple data sources explicit.

We also feel that the metadata-based approach can be cost-effective. Using metadata we are better able to understand the structure and the semantics of the available data and hence make better use of that data. It is also true that metadata changes much less frequently than its associated data. This means that it can be cost-effective to put effort into extracting metadata since it has a relatively long lifetime.

The most important part of the approach is the design of the metadata schema since it must capture the important properties and relationships of the data and it is used as the basis for the extraction and query tools. We believe that an object-oriented model is preferable for two reasons. First, the concepts of object and class are applicable to a wide variety of data sources, that is they are able to represent the metadata for these data sources. Second, the schema is scaleable in the sense that we can easily introduce more types of data sources into a schema by extending the class hierarchy with the *is-a* relationship.

While the extraction tools for the two examples were built specifically to process their respective types of data sources there are similarities. In particular, both tools base the data processing on the metadata schema for the example. Thus it should be possible to construct a general tool that can be customized for each situation. We see this general tool consisting of a parser component for processing the data from the passive data sources, and a library of routines, based on the MDR Client Library, for placing the metadata into the MDR. The parser would have to change for different types of data sources but the library of access routines would be the same for all instances of the tool. The robot architecture of the Web metadata collection tool discussed in Section 4.2 could be used for any collection of data sources on the Web.

The processing performed by each component of the general extraction tool is guided by the metadata schema developed for the collection of data sources. The schema, therefore, would be the main input to the tool creation process. We believe that tool creation could be automated to a large degree by adapting techniques used in parser generation, for example the structural transformation language TXL [6]. We have had success in using structural transformation to perform schema translation [1] and believe that this would be a reasonable approach. Using a language like TXL for this purpose would require us to specify the syntax for both the metadata schema and the data. Generalizing our approach would also require that we standardize class definitions to some degree.

The query tools are more general than the metadata extraction tools since they interact with the MDR and not with the data sources. The MDR Browser discussed in Section 4.1 is already a very general tool. It will support browsing and querying with any metadata schema. The Web query tool discussed in Section 4.2 is more specialized than the browser because it has an interface customized to the Web application. The query engine, however, is general and would work with any schema. Thus the Web query tool can be customized to particular collections of data sources on the Web by tailoring the query interface to the particular metadata schema.

6. Conclusions

The growth of the Internet has meant that there is now an abundance of information available in passive data sources, that is data sources which must be searched and accessed by “third-party” techniques and tools. A key problem being addressed by a number of researchers is how to make effective use of this data.

In this paper we outlined an approach to the problem of querying passive data sources which is based on the extraction, and subsequent exploitation, of metadata from the data sources. The main task in the approach is the design of a metadata schema which captures the high-level structure and relationships present in the data. The schema is then used as input to the construction of tools to extract metadata from the data sources and store it in a repository, and to query the metadata once in the repository. Users can conveniently query the metadata in order to locate data of interest within the collection of passive data sources.

Our metadata-based approach has several advantages. First, the approach is applicable to any type of passive data source. It is particularly useful for text data sources since we can build automated, or at least semi-automated, metadata extraction tools. Second, the approach is able to integrate the metadata for a single concept in the data where data about that concept may be in multiple data sources. Third, the approach is cost-effective. Using metadata we are better able to understand the structure and the semantics of the available data and hence make better use of that data. It is also true that metadata changes much less frequently than its associated data so the cost of extracting metadata can be amortized over a relatively long lifetime. Fourth the approach is scaleable in the sense that more data sources may be easily added as long as they conform to the existing schema.

We described a metadata model which is used to develop the metadata schema. The model is structurally object-oriented and supports powerful abstraction mechanisms like generalization and classification. The model has two main advantages. First, it can capture the metadata descriptions for a variety of data sources. Second, the schemas produced with the model are scaleable in the sense that we can easily introduce more types of data sources into a schema by extending the class hierarchy with the *is-a* relationship. However, adding new classes to a schema will force changes to the associated data extraction tool since it must recognize the new classes. We also describe a prototype metadata repository which was built to support the research.

We presented two example applications in which the approach was used. The first example uses metadata about application files to support configuration management of distributed applications. The second example uses metadata to support querying documents on the World Wide Web. In both examples, a metadata schema was developed to represent the metadata and tools were developed to extract the metadata from the data sources and to query the metadata once it was placed in the MDR.

Based on our experiences we conclude that our approach can be generalized to handle other types of passive data sources. The metadata model, as we discussed above, is capable of representing other situations. We also believe that the tools we have built are either general enough now or can be made more general. The MDR Browser is a general tool that can be used with any metadata schema. The Web query tool can be adapted by replacing the query interface to match a new schema. We suggest that data extraction tools can be derived from a general tool consisting of a parser component for processing the data from the passive data sources, and a library of routines for placing the metadata into the MDR. The parser would have to change for different types of data sources but the library of access routines would be the same for all instances of the tool. The parser component for a particular situation could be created with a parser generator such as TXL.

References

[1] R. Abu-Hamdeh, J. Cordy and P. Martin, "Schema Translation Using Structural Transformation", *Proc. of CASCON' 94: IBM Centre for Advanced Studies Conference*, Toronto ON, November 1996, pp. 202-215.

[2] G. Attaluri, D. Bradshaw, N. Coburn, P.-A. Larson, P. Martin, A. Silbershatz, J. Slonim and Q. Zhu, "The CORDS Multidatabase Project", *IBM Systems Journal 34(1),* 1995, pp. 39 - 62.

[3] M. Bauer, R. Bunt, A. Rayess, P. Finnigan, T. Kunz, H. Lutfiyya, A. Marshall, P. Martin, G. Oster, W. Powley, J. Rolia, D. Taylor and M. Woodside, "Services Supporting Management of Distributed Applications and Systems", To appear in *IBM Systems Journal*.

[4] C. M. Bowman, P.B. Danzig, U. Manber and M.F. Schwartz, "Scaleable Internet Resource Discovery: Research Problems and Approaches", *Comm. of the ACM 37(8),* August 1994, pp. 98 - 107.

[5] C. M. Bowman, P.B. Danzig, U. Manber and M.F. Schwartz, "The Harvest Information Discovery and Access System", *Proceedings of the 2nd World Wide Web Conference*, 1994, pp. 763 - 771.

[6] J. Cordy, C. Halpern-Hamu and E. Promislow, "TXL: A Rapid Prototyping System for Programming Language Dialects", *Computer Languages 16(1)*, 1991, pp. 97-107.

[7] O. Etzioni, "The World Wide Web: Quagmire or Gold Mine?", *Comm. of the ACM 39(11),* November 1996, pp. 65- 68.

[8] C. Hsu, M. Bouziane, L. Rattner and L. Yee, "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach", *IEEE Transactions on Software Engineering 17(6),* June 1991, pp. 604 - 625.

[9] W. Klas and A. Sheth (Editors), "Special Issue: Metadata for Digital Data", *SIGMOD Record 23(4),* December 1994.

[10] D. Konopnicki and O. Shmueli, "W3QS: A Query System for the World Wide Web", *Proceedings of the 21st VLDB Conference*, 1995, pp. 54 - 65,.

[11] M. Koster, *The Web Robots Database 1997*. From http://info.webcrawler.com/mak/projects/robots/active.html, downloaded March 1997.

[12] A. Levy, A. Rajaraman and J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions", *Proceedings of the 22nd VLDB Conference*, Mumbai India, 1996

[13] P. Martin and W. Powley, "An Information Model for Distributed Applications Management", *Proc. of CASCON' 96: IBM Centre for Advanced Studies Conference*, Toronto ON, November 1996, pp. 54 - 63,.

[14] P. Martin and W. Powley, "Catalog Management in Multidatabase System using an X.500 Directory System", accepted for publication in *Distributed Systems Engineering Journal*, May 1997.

[15] A. Mendelzon, G. Mihaila and T. Milo, "Querying the World Wide Web", *Proceedings of PDIS'96*, Miami FL, 1996.

[16] J. Mylopoulos, A. Borgida, M. Jarke and K. Koubarakis, *Telos: A Language for Representing Knowledge about Information Systems (revised)*, Technical Report KRR-TR-89-1, Department of Computer Science, University of Toronto, August 1990.

[17] OMG, "Common Object Request Broker Architecture: Architecture and Specification", *OMG Document No. 91.12.1*, 1991.

[18] OSF, *The OSF Distributed Computing Environment Rationale*, Open Software Foundation, Cambridge MA, 1991.

[19] A. Weston, *Using a Data Model to Search and Query the World Wide Web*, Master's thesis, Department of Computing and Information Science, Queen's University, 1997.

[20] B. Yuwono and D.L. Lee, "WISE: A World Wide Web Resource Database System", *IEEE Transactions on Knowledge and Data Engineering 8(4),* August 1996, pp. 548 - 554.