IAIS: A Methodology to Enable Inter-Agency Information Sharing in eGovernment

Akhilesh Bajaj, Carnegie Mellon University, USA
Sudha Ram, University of Arizona, USA

ABSTRACT

Recently, there has been increased interest in information sharing among government agencies, with a view toward improving security, reducing costs and offering better quality service to users of government services. In this work, the authors complement earlier work by proposing a comprehensive methodology called IAIS (Inter Agency Information Sharing) that uses XML to facilitate the definition of information that needs to be shared, the storage of such information, the access to this information and finally the maintenance of shared information. The authors compare IAIS with two alternate methodologies to share information among agencies, and analyze the pros and cons of each. They also show how IAIS leverages the recently proposed XML (extensible markup language) standard to allow for inclusion of various groups’ viewpoints when determining what information should be shared and how it should be structured.

Keywords: XML, agents, egovernment, information sharing, heterogeneous databases

INTRODUCTION

The emergence of the Internet and its applications has fundamentally altered the environment in which government agencies conduct their missions and deliver services. Recently, there has been considerable interest in exploring how emerging technologies can be used to promote information sharing among different governmental agencies. Such information sharing is desirable for several reasons. First, increased levels of security can be achieved if different government agencies share information. These effects can be felt in areas as diverse as global counter-terrorism (Goodman, 2001), homeland security (Rights, 1984) and the war on drugs...
(Forsythe, 1990). Several recent articles, (e.g., Dizard, 2002) strongly endorse the view that the sharing of intelligence information amongst different law enforcement agencies will enhance their ability to fulfill their required functions. Second, there has been a growing need to streamline interagency communication from a financial savings perspective. For example, Minahan (1995) shows how the lack of information sharing between different government organizations considerably hampered the establishment of an import-export database that would have streamlined the flow of goods into and out of the US and potentially saved billions of dollars. As pointed out in Stampiglia (1997), data sharing between health care agencies can also result in significant cost savings. Third, interagency information sharing results in offering fewer contact points for end-users of public services, thereby leading to more efficiencies in the delivery of these services to the end-users. For example, allowing agencies to share geographic information systems (GIS) information improves the quality of customer service afforded to end-users of these services (Hinton, 2001).

Other common examples of activities that can benefit from information sharing include: the application for licenses for business expansion, and the ability of aid workers to provide services such as home delivered meals and in-home care.

The benefits of information sharing have to be weighed against concerns about potential privacy violations, which preclude the establishment of a single database that can be accessed by multiple agencies. This has been pointed out in several areas such as health care (Gelman, 1999), electronic voting (Hunter, 2002), and public life in general in the post-September 11, 2001 world (Raul, 2002). Given the tradeoffs between information sharing and privacy, it is well accepted that multiple players need to be involved when determining what information should be shared. These players may include: a) privacy advocacy groups such as the Privacy Rights Clearinghouse (www.privacyrights.org), b) government agencies involved with producing, sharing, or using the shared information, such as law enforcement agencies, and c) legislative and executive bodies that formulate and execute legislation for information sharing in different instances.

A considerable body of work exists in the area of the integration of structured information between heterogeneous databases (Hayne & Ram, 1990; Reddy, Prasad, Reddy, & Gupta, 1994; Larson, Navathe, & Elmasri, 1989; Batini, M.Lenzerini, & Navathe, 1986; Hearst, 1998; Ram & Park, 2003; Ram & Zhao, 2001). The two broad approaches in this area are a) the creation of virtual federated schemas for query integration (Zhao, 1997; Chiang, Lim, & Storey, 2000; Yan, Ng, & Lim, 2002) and b) the creation of actual materialized integrated warehouses for integration of both queries and updates (Vaduva & Dittrich, 2001; Hearst, 1998). While the area of structured information integration is relatively well researched, considerably less attention has been paid to the area of the integration of unstructured information (e.g., free text documents) between heterogeneous information sources. Recently, several researchers e.g., (Khare & Rifkin, 1997; Sneed, 2002; Glavinic, 2002) have pointed out the advantages of the XML (extensible markup language) standard as a means of adding varying degrees of structure to information, and as a standard for exchanging information over the WWW.

As pointed out in Dizard (2002), Minahan (1995), and Stampiglia (1997), much of the information that government
organizations share is at least somewhat unstructured. The primary contribution of this work is a comprehensive methodology that we call IAIS (inter-agency information sharing) that enables information sharing between heterogeneous government organizations. IAIS leverages the XML standard and allows for: a) the ability to provide varying degrees of structure to the information that needs to be shared, by sharing all information in the form of XML documents; and b) the inclusion of various groups’ viewpoints when determining what information should be shared and how long it should be shared. Once the structure of the information to be shared has been determined, IAIS utilizes a novel method of storing and accessing the information. In this work, we describe IAIS and utilize well-understood criteria such as ease of information definition and storage, ease of information access, and ease of system maintenance to compare IAIS to alternate methodologies of information sharing.

The rest of this paper is organized as follows. In the next section, we discuss prior research in the area of information integration and position IAIS in that context. Then we present the IAIS methodology and alternative strategies for information definition as well as mechanisms for information storage and retrieval and compare IAIS to these alternate methodologies. A conclusion and future research directions complete this paper.

PREVIOUS WORK

Earlier work in the area of information integration has focused primarily on integrating structured data from heterogeneous sources. Excellent surveys of data integration strategies are presented in Batini et al. (1986), Hearst (1998) and Chiang et al. (2000). There have been primarily two broad strategies used to integrate structured data: a) retain the materialized data in the original stores, but use a unified federated schema to allow the querying of heterogeneous sources; or b) actually materialize the combined data into a unified repository to allow for faster query response and also allow updates. Note that b) requires a unified federated schema also, but the schema in this case is not virtual, as in a).

Work in the area of unified federated schema generation is well established (e.g., Batini et al., 1986) has an excellent survey of early work in the area). Several issues have been addressed in this area. First, Blaha & Premerlani (1995) highlight the commonly observed errors in the design of the underlying heterogeneous relational databases, which often need to be resolved before integration is possible. Second, the issue of semantic inconsistencies between database object names (such as attribute names) has been widely addressed. Strategies to resolve semantic conflicts have ranged from utilizing expert systems (Hayne & Ram, 1990) to neural networks (Li & Clifton, 1994). Recently, several researchers have recognized this problem to be only partially automatable, e.g., (Chiang et al., 2000). A recent solution to partially automate semantic resolution in Yan et al. (2002) utilizes synonym sets, with similarity measures. A set of potential unified federated schemas is generated using algorithms proposed in this work, and final selection of the unified schema is done manually. As an alternate solution, a schema coordination methodology is proposed in Zhao (1997), where the minimal mapping is done only at the semantic level (rather than at the logical level), and overheads are lower than in traditional schema integration. However the tradeoff is that this methodology can only be used for querying, and the query
resolution process is more complex than with a federated schema.

In the area of materialized data integration, a federated schema is required as a first step. However, there are several additional issues such as the retention of legacy systems, the coordination and refresh rate of information in the materialized warehouse, and the resolution of data quality issues (Chiang et al., 2000; Vaduva & Dittrich, 2001). Chiang et al. (2000) highlight possible problems that can arise when integrating actual data, after the schema integration has taken place. Examples of these problems include entity identification, relationship conflicts and attribute value discrepancies between data from heterogeneous sources. Even though much work has been done in the area of the integration of structured information, Silberschatz, Stonebraker, and Ullman (1996) highlight it as one of the major research directions for database research in the future, and work continues in this area.

Unlike the integration of structured information, considerably less work has been done in the area of unstructured information integration. The domain of interest in our work is the sharing of information between government organizations. This raises several new issues. First, while traditional integration work has considered domains where there are heterogeneous structured database schemas, much of the information shared between government organizations tends be unstructured (Dizard, 2002; Minahan, 1995; Stampiglia, 1997). As such, structured data integration methodologies are insufficient to allow data sharing between government organizations. Second, because information in government organizations is often sensitive from a policy and privacy perspective, the actual definition of the information that needs to be shared is often performed by several parties. Thus, applicable methodologies in our domain of interest should allow various groups to flexibly structure information at varying degrees of structural rigidity. Thus, for example, certain information (such as names and addresses) can be structured down to the same detail as relational database columns are structured, while other information (such as descriptive comments, rules and regulations) should be retained as free text. Our work complements work in the area of structured data integration by proposing a methodology that satisfies these additional requirements.

Recently, several researchers have pointed out the advantages of the XML standard as a means of adding varying degrees of structure to information, and as a standard for exchanging information in different domains. For example, Sneed (2002) points out how XML can be used to pass data between different software programs in batch or real-time mode. Glavinic (2002) describes how XML can be used to integrate applications within an organization. In the IAIS methodology described in this work, we leverage XML in order to allow the sharing of information between heterogeneous information sources, regardless of the underlying data sources. This is advantageous since in a government organization, information sources can range from repositories of text documents to relational databases.

IAIS: INTER-AGENCY INFORMATION SHARING

Core Design Criteria Behind IAIS

There are several core design criteria that underlie IAIS. First, IAIS facilitates the definition of the structure of the information that is to be shared between agencies. This allows the different players men-
tioned earlier, such as privacy advocacy groups, governmental agencies and legislative committees, to learn the methodology once and apply it for each instance of information sharing, regardless of which agencies are involved. Since the same bodies are likely to be involved in several instances of sharing (across different agencies) this is an important criterion. Second, it is important that IAIS be easy to maintain without requiring a significant increase in information system (IS) maintenance costs. Since many governmental agencies have budgetary constraints that prohibit increased hiring of costly IS personnel, this is an important constraint. Consequently, IAIS trades off optimal efficiency in searching for ease of storage and maintenance. Third, given the rapidly increasing amount of digital information in government organizations (Dizard, 2002), IAIS is designed to be scalable, so that it can be used to share large amounts of information. Fourth, we designed IAIS to be easy to use when searching for information.

Components of IAIS

IAIS consists of three major components: a) an information definition component, b) an information storage component, and c) an information retrieval component.

The first component, i.e., the information definition component, is used when different agencies agree to share information (an information sharing instance). The first step is for the players involved in an information sharing instance to create Document Type Definitions (DTD) of the information to be shared, and also set limits on the amount of time information items are shared. For example, if a county’s police and treasury departments wish to share information, they will first collaborate and create DTDs of the information items they wish to share. These DTDs will be discussed and agreed upon by the various players described earlier, until a settlement is reached. An example of two such DTDs are shown in Figure 1. The DTD in Figure 1a shows information that the police agree to provide to the treasury department if the latter desires to verify if a specific business license applicant has been convicted of a felony. The information on each felon contains the name (first and last), the social security number, a list of convictions, and an expiry date. The convictions list has repeatable conviction items. PCDATA stands for parsed character data (text) while the * symbol indicates that the item can be repeated. The DTD in Figure 1b is information that the treasury department agrees to share with the police to notify them of business applications that are being processed. The <expirationdate> tag allows a software program to delete elements whose expiration date has passed.

The second component, i.e., the information storage component, is used to store the actual XML pages that conform to these DTDs. First, a one-time effort is required to create reports from the local databases or document repositories (of the police and the treasury) in the form of XML pages. Since most commercial DBMSs currently offer the ability to create scheduled reports in XML, this task is possible without adding to existing IS systems and personnel. Next, a repository is created to store these pages. This repository is a directory structure with an IP (Internet protocol) address that consists of interlinked HTML and/or XML pages. Since most commercial DBMSs currently offer the ability to create scheduled reports in XML, this task is possible without adding to existing IS systems and personnel. Next, a repository is created to store these pages. This repository is a directory structure with an IP (Internet protocol) address that consists of interlinked HTML and/or XML pages. Finally, procedures are implemented to affect the periodic transfer of pages to this repository, and purging of these pages from the repository.

The third component, the information retrieval component, is a search agent called XSAR (XML search agent for in-
information retrieval) that searches for information from the repository (Bajaj, Tanabe, & Wang, 2002). XSAR can dynamically query large information repositories of XML documents. It is DTD independent, so the same agent can be used to query documents consisting of multiple DTDs. XSAR is a dynamic agent, in that it does not use an underlying database or directory scheme of information on pages. Rather, it dynamically queries the repository on behalf of a user. There is no requirement for the repository site to have any special software or hardware; the only assumption is that it is a site that contains a mixture of HTML and XML pages.

XSAR does require the user to be aware of the underlying DTD, insofar as being aware of which fields the user needs to search. Essentially, XSAR reformulates the query as an XQL expression, and then launches a spider that traverses the information repository and executes the XQL query against each XML page found in the repository. Figure 2 shows the operational flow of XSAR.

Figure 1. Example of DTDs created to share information between Police and Treasurer.

```
<?XML version = "1.0" ?>
<!DOCTYPE DOCUMENT [
  <!ELEMENT DOCUMENT (FELON)*>
  <!ELEMENT FELON (NAME, SSN, CONVICTIONS,EXPIRATIONDATE)>
  <!ELEMENT NAME (LASTNAME,FIRSTNAME)>
  <!ELEMENT LASTNAME (#PCDATA)>
  <!ELEMENT FIRSTNAME (#PCDATA)>
  <!ELEMENT SSN(#PCDATA)>
  <!ELEMENT CONVICTIONS (CONVICTIONITEM)*>
  <!ELEMENT CONVICTIONITEM (CONVICTIONDATE,CONVICTIONDESCRIPTION,CONVICTIONSENCE)
   <!ELEMENT CONVICTIONDATE (#PCDATA)>
   <!ELEMENT CONVICTIONDESCRIPTION (#PCDATA)>
   <!ELEMENT CONVICTIONSENCE (#PCDATA)>
   <!ELEMENT EXPIRATIONDATE (#PCDATA)> ]>

a) DTD of Information Shared from Police to Treasurer.

<?XML version = "1.0" ?>
<!DOCTYPE DOCUMENT [
  <!ELEMENT DOCUMENT (APPLICANT)*>
  <!ELEMENT FELON (NAME, SSN, APPLICATIONS,EXPIRATIONDATE)>
  <!ELEMENT NAME (LASTNAME,FIRSTNAME)>
  <!ELEMENT LASTNAME (#PCDATA)>
  <!ELEMENT FIRSTNAME (#PCDATA)>
  <!ELEMENT SSN(#PCDATA)>
  <!ELEMENT APPLICATIONS (APPLICATIONITEM)*>
  <!ELEMENT APPLICATIONITEM (APPLICATIONDATE,APPLICATIONDESCRIPTION,APPLICATIONDECISION)>
  <!ELEMENT APPLICATIONDATE (#PCDATA)>
  <!ELEMENT APPLICATIONDESCRIPTION (#PCDATA)>
  <!ELEMENT APPLICATIONSENCE (#PCDATA)>
  <!ELEMENT EXPIRATIONDATE (#PCDATA)> ]>

b) DTD of Information Shared from Treasurer to Police
```
Prototype Implementation of IAIS

Our implementation of XSAR was built using the Java Server Pages (JSP) architecture, which is part of the J2EE (Java 2 Enterprise Edition) specification, accessible currently at http://java.sun.com. This architecture has the advantage of cross-platform portability, in that the agent software is operating system independent, and the search results produced (JSP pages) are browser independent. Figure 3 illustrates the high-level architecture of XSAR as per the Model View Controller (MVC) architecture (Foley & McCulley, 1998).

Choice of XML Query Language

Currently, there is no standard query language for XML designated by the WWW Consortium (W3C). However, several query languages for XML documents have already been proposed, e.g., XQL, XML-QL and Quilt (all accessible via http://www.w3c.org). We selected XQL for the implementation because: a) its grammar is based on Xpath, which has already been standardized by W3C; and b) application programming interfaces (API) for XQL are available in Java.

XSAR offers several improvements over existing facilities for searching XML files. First, XQL by itself can be used to query only individual XML pages, and to return results from that page. It does not query multiple XML pages, and return combined results from them. XSAR overcomes this limitation by crawling through the repository, and executing the XQL query on each XML page. XSAR combines and displays the results of its repository search to the user. Second, while XSAR currently uses XQL, the source code underlying XSAR can be easily modified to use a different API, to accommodate future changes in the W3C specifications. Third, XSAR allows the use of XQL to be transparent to the user, by providing the option of an easy-to-use forms-based interface (see Appendix 1, Figure A1-1) to formulate the query.

Functionality of XSAR

Based on the operational description of XSAR (see Figure 2), detailed
screenshots of actual XSAR usage are shown in Appendix 1. The case in the appendix searches a repository that lists felon information, as shown in our example in Figure 2. It shows a search for a specific suspected felon whose last name is ‘Smith’ and first name is ‘John’. There are two modes in which users can interact with XSAR. In the expert mode, the user formulates queries using XQL, while in the novice mode she can use the menus for searching. In Figure A1-1, the novice mode is used for specifying search conditions. Single quotes are used to surround string parameters. If a parameter is a number, no quotation is required. Each comparison operator is of two types: case sensitive and case insensitive. Figure A1-2 in Appendix 1 shows the search result of the query posed by the user. In this case, only one XML document was found. Figure A1-3 in Appendix 1 shows the result of selecting the “See Matches” option in Figure 2. Only the matched part of felons.xml is displayed. Figure A1-4 in Appendix 1 shows the result of choosing the “See whole XML” option in Figure A1-2. The entire felons.xml is displayed.

Detailed Architecture of XSAR

As illustrated in Figure 3, the underlying architecture of XSAR follows the Model View Controller architecture which differentiates between the data, the program logic and the viewing interface (Foley & McCulley, 1998). XMLRetriever and ShowMatch are the Controller: they are Java Servlets which are responsible for receiving the requests from the client. The Agent component takes responsibility for retrieving the data from the website.
(Model). The search results are presented as JSP pages (View).

Once XMLRetriever receives the request from the client, it generates the XML query string and hands it to Agent. Agent essentially crawls through the target repository, utilizing a breadth-first search algorithm. It parses the starting page (specified by the user) and identifies links. Each linked page is then parsed (if it is an HTML page) or parsed and searched (if it is an XML page). ShowMatch gets the search results from the agent program and puts them into JSP files for presentation.

XSAR utilizes the Xerces API for XML page parsing, the GMD-IPSI XQL engine for XQL querying and is packaged to run on the Tomcat web container. Appendix 2 flowcharts the high level logic of each of the software components used in XSAR. Next, we describe alternate methods for information definition, storage and access and compare IAIS with alternate potential methodologies that can be used to facilitate information sharing in our domain of interest.

COMPARING IAIS WITH OTHER POTENTIAL METHODOLOGIES

Alternative Strategies for Information Definition, Storage and Access

Information Definition

Several choices are available for specifying how the information to be shared among government agencies can be defined. We list three commonly used standards here. The first and most obvious is free text documents, where the name of the document along with other fields such as keywords can be used to structure information about the document. This standard includes documents in pure text markup languages such as HTML (hypertext markup language). Free text documents are appropriate when storing records of proceedings, hearings, rules and regulations or memoranda. A significant portion of government information is in this format (Dizard, 2002; Minahan, 1995; Stampiglia, 1997).

The second standard is using a relational database structure (Codd, 1970), and it involves storing information in the form of relations, where each table has attributes, and the tuples contain the actual data. Relations are linked to each other using common attributes that usually uniquely identify a tuple in one table. The relational model has well-accepted rules for eliminating data redundancy using the concept of data normalization. Most commercial database management systems (DBMSs) support this standard. This option is useful if the information to be shared consists of clearly defined attributes such as: the name, address, physical description and attributes describing immigration entries and exits for individuals.

The third standard for structuring information is the use of XML DTDs. XML DTDs allow information to be structured, using tags to differentiate elements and sub-elements. This is analogous to tables and columns in the relational model. Several researchers believe that XML will allow the Web to become more semantic in nature, such that software agents will be able to access information and understand what it means (Berners-Lee, Hendler, & Lassila, 2001). The main advantages of XML are: a) it is a platform independent standard where the data does not need to be read by a proprietary product (as it would in a relational DBMS), and b) it allows for variable structuring of information, where some parts can be free text and others can be more structured.

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
Information Storage and Access

Based on the three alternate standards of information definition described earlier, there are alternate methods of storing the structured information as well. If the information is structured in the form of free text or HTML files, then storage on a file server (usually a component of the operating system) is a first alternative. Another option is third-party document management tools such as Documentum (www.documentum.com) that facilitate the creation and management of these pages. Most of these systems use the existing file system to store the applications, but store additional information that makes it easier to retrieve the files. Finally, it is possible to use object-relational features of current commercial DBMSs to store and retrieve these files.

For information stored in the form of free text, search engines, e.g., Google and AltaVista, offer one method of accessing information. This is similar to a search on the WWW, except that instead of searching across numerous web sites, the search in our case is on a repository of free text or HTML documents. A second method is to use directory-based search engines such as Yahoo (www.yahoo.com). If a third-party application is used for storage, the search capabilities are limited to the capabilities of the tools offered by that application. Finally, if object-relational DBMSs are used, the ease of access depends on extensions to the structured query language supported by the DBMS.

If the information is structured in a relational database, a commercial DBMS can be used to store this information. Commercial relational DBMSs are a tested technology that allow for concurrent access by multiple users. They provide good throughput, recovery and reliability for most applications. The structured query language (SQL) standard allows easy access to information. Results can be served on the WWW using a number of available technologies such as active server pages (ASP) or JSP.

If the information is structured in XML, it consists of text files, though the text is now structured. Storage alternatives for XML files are similar to free text, i.e., file systems, third-party applications and object relational DBMSs. However, there is an important difference between free text and XML formats, when accessing the information. For XML files, the methodology of using conventional search engines with keyword searches either breaks down, or at the very least negates the whole purpose of structuring the information as an XML document. Similarly, directory-based search engines do not allow the search and retrieval of portions of XML pages. To overcome these problems, two standards have been proposed by the WWW consortium (www.w3c.org) to search XML files. These are the XPATH standard, and the more recent XQUERY standard, which is still evolving. As of now, these standards allow the querying of a single XML page, not a repository of XML pages.

Figure 4 summarizes the different methods of information storage (in each column), and the alternate methods available to access them (along the rows).

Based on the above alternatives, we compare IAIS in this work with two alternate methodologies. In the first methodology, called the free text methodology, the parties involved make a listing of the free text documents the government agencies should share. These documents are then stored in a free text /HTML format in a centralized repository and searched by end users using keyword based search engines such as Google. The second methodology, called the database schema methodology, requires that the various parties examine
the relational schemas of the underlying databases of the different agencies whose information is to be shared. They decide what information needs to be made available from each of these databases. This information can then be searched via a single Web interface that queries the individual databases directly, though the data itself will be stored in the individual databases.

Next, we compare these methodologies with IAIS along the following dimensions: a) ease of information definition and storage, b) ease of information access, and c) ease of system maintenance.

Ease of Information Definition and Storage

In the free text methodology, unstructured information such as agency documents can be identified and shared. However, it is not possible to define specific, structured information that can be shared. Since most agencies have a collection of documents, the information definition step will in general be the easiest in this phase, and consist of a relatively simple identification of documents that can be shared, without any in-depth study of the actual data elements in the agencies. In the database schema methodology, the schema of each government agency will have to be prepared for review by the different parties. Different data items can then be identified for sharing. A similar process can be followed for IAIS, where the data elements will need to be defined and then a DTD created. Thus, both the database schema methodology and the IAIS methodology require an examination of the actual data elements used by the agencies.

From the storage standpoint the free text methodology requires that the pages be stored on a server’s file system. A coordination mechanism is needed in place so that changes to documents in the organizations’ internal repositories are reflected in the shared repository. In the database schema methodology, there is no need for additional storage of data, since information is obtained dynamically from the underlying databases within each organization. In IAIS, the XML pages are stored in a repository, similar to the free text methodology, with a coordination mechanism to ensure that the pages are refreshed periodically to reflect the structured information within the internal databases. Thus, the database schema methodology requires less implementation effort from a storage perspective, than the free text and the IAIS
methodologies.

From this discussion, in general, we see that the free text methodology offers the easiest data definition while the database schema methodology offers the easiest information storage.

**Ease Of Information Access**

Web search engines that search files on different Web sites have their origin in the information retrieval (IR) systems developed during the last 50 years. IR methods include Boolean search methods, vector space methods, probabilistic methods, and clustering methods (Belkin & Croft, 1987). All these methods are aimed at finding documents relevant for a given query. For evaluating such systems, recall (ratio of the number of relevant retrieved documents to the total number of available relevant documents) and precision (ratio of the number of relevant retrieved documents to the number of total retrieved documents) are the most commonly used measures (Billsus & Pazzani, 1998). Finally, response time (time taken for the user to get the information she desires) has been found to be a useful metric (Anderson, Bajaj, & Gorr, 2002).

Furthermore, the performance of each search tool is directly influenced by the user’s ability to narrowly define the nature of the query, e.g., in the form of precise keyword strings or correct specification of a search condition. For the sake of the discussion below, we assume that the search specificity is held constant.

**Recall**

For the free text methodology, recall is the ratio of the number of pages identified correctly as matching to the total number that should have been identified as matching. This ratio is clearly dependent on the algorithms used to perform the matching, with most current algorithms based on weighted indices. In general the ratio will be less than 1. Also, as the number of pages increases, all else being equal, the recall drops.

For the database schema methodology, the recall of a search is determined by the accuracy of translation of the user’s query into the underlying database language (usually SQL) query. If the query translation is accurate, and the underlying database is searched completely, the recall will be 1.

For agents like XSAR (the search mechanism in IAIS), the recall is dependent on the accuracy of translation of the user query into the underlying XML query language, and the existence of a path from the specified starting point to all the XML pages that exist in the repository. For a well-connected repository where every page is reachable from a starting node, the recall will be 1. From this discussion, we can see that in general, the recall for the free text methodology would be lower than for the other two methodologies.

**Precision**

For the free text methodology, precision is determined by: a) the features offered to the user to specify a query to a desired level of precision (e.g., concatenated strings, wild card characters, boolean operators); b) the extent to which precision can be sacrificed for response time; and c) the extent of information captured in the underlying search engine database about the real pages. All else being equal, a search engine that offers more options for searching should have greater precision, as compared to one where response time is secondary to precision. For c), a search engine that captures only the title of each page will have less precision than
one that captures the title and the META tags of each page. As the number of pages increases, all else being equal, the precision drops.

When using the database schema methodology, the precision of an information repository that uses an underlying database depends on: a) the features provided by the user query interface and b) the maximum precision allowed by the underlying query language of the database. If a relational database is used, then the underlying query language is SQL, which provides excellent precision.

For XSAR, the query interface fully supports XQL. Thus, the precision of XSAR is identical to the precision of XQL, which supports XPath (http://www.w3c.org), the W3C standard for precision.

Response Time:
The response time in the free text methodology depends on: a) the algorithm used to perform the keyword match with the database that has information on each page in the repository, and b) the extent to which the designers are willing to sacrifice precision for response time.

When using the database schema methodology, the response time depends on the complexity of the specified query and the extent to which the database is optimized for that query. Thus, for a relational database, a range query on non-indexed attributes with several joins is likely to take significantly more time than a simple SELECT query. Thus, these repositories would need a database administrator to keep track of most frequent queries, and ensure proper tuning access optimizations.

For agents like XSAR, response time is determined by the time taken by the agent to crawl through the target repository. This depends on exogenous factors such as network speed, performance of the web server of the target repository, and the size of the target repository. In designing XSAR, we used three endogenous strategies to minimize this time, for given values of exogenous factors: a) a multi-threaded agent, b) a proxy server for caching and c) providing the user the ability to specify response time threshold. We next describe each of these strategies.

As the agent program fetches new pages from the target repository, it spawns threads to parse these fetched pages. HTML pages are only parsed for links, while XML pages are parsed and searched for the query. One design trade-off here is that spawning a new thread is expensive if the pages being parsed and/or searched are small, while increasing the number of threads pays off as the size of each page in the repository increases. After experimenting with different real XML repositories (see Appendix 2), we selected five threads per search for the current implementation. However XSAR is an open source software, and this number can be modified under different installations.

For proxy caching in XSAR, we used the DeleGate proxy server, developed by Yutaka Sato, Japan (www.delegate.org/delegate/). A proxy server is useful in reducing network traffic, since it caches frequently accessed pages, and hence reduces an agent’s (or browser’s) requests to a distant WWW server. We tested the performance of using the proxy server for a set of cases. The results of using the proxy server are shown in Appendix 2.

Finally, XSAR allows the user to set the threshold response time and maximum depth to which the agent should search the target repository. This allows XSAR to be used by different classes of users, e.g., one class of users may want to search large repositories comprehensively, and leave the agent running, say, overnight, while another
class of users may want quicker responses for searches where the maximum depth of the search is set to a finite number.

We tested the performance of XSAR using three experiments described in Appendix 3. In our experiments, a multi-threaded program had the maximum impact in reducing response time, with a proxy cache playing a secondary role. The results (see Appendix 3) indicate that XSAR is clearly slower than the other mechanisms available for searching, which is to be expected, since it searches a target repository dynamically. XSAR trades off faster response time for the ability to search a changing information repository in real time.

Figure 5 summarizes the performance comparison between the four search mechanisms.

**Ease Of System Maintenance**

In the free text methodology, the major maintenance issues include the upkeep of the document repository, the search engine and the coordination software required to maintain consistency between the internal pages within the organization and the pages in the information repository. In the database schema methodology, there is a need to track changes to the underlying database schema. Whenever these changes to the schema occur, appropriate changes are needed in the application code that directly queries the database. Furthermore, the organizations have to reveal their schemas to the team maintaining the application code in the system used for information sharing. In the IAIS methodology, there is an added layer of transparency, since each organization promises to deliver information in the form of XML pages conforming to the DTDs. This allows each organization to hide their internal database schemas and to make schema changes independent of the information sharing process, as long as each organization delivers its required pages. The maintenance issues for the IAIS information sharing system include refreshing pages from each organization and maintenance of the search agent (such as XSAR).

From the above discussion, we can see that in general the database schema method requires the most maintenance, while the free text and IAIS methods require less maintenance.

The results of our comparison of these three methodologies are shown in Figure 6.

From Figure 6, it is clear that the main advantages of using IAIS over the free text methodology are a) the ability to share structured information and b) the ability to search a repository dynamically with greater precision and recall. The main advantages of IAIS over the database schema methodology are: a) the former is

---

**Figure 5: Comparison Of Methods For Information Access**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Recall</th>
<th>Precision</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Text Method</td>
<td>Low</td>
<td>Low</td>
<td>Fast</td>
</tr>
<tr>
<td>Database Schema Method</td>
<td>100%</td>
<td>High</td>
<td>Fast</td>
</tr>
<tr>
<td>IAIS</td>
<td>100%</td>
<td>High</td>
<td>Slow</td>
</tr>
</tbody>
</table>

---

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
in general easier to maintain and precludes the need for making database schemas available across organizations, because of the increased layer of indirection provided by DTDs; and b) the ability to share unstructured as well as structured information in IAIS. The tradeoff is that queries in IAIS require more response time than in the other methodologies.

**CONCLUSION AND FUTURE RESEARCH**

In this work, we proposed IAIS: an XML-based methodology to facilitate sharing of information among government agencies. Information in this domain tends to be both structured and unstructured. IAIS complements previous work in the integration of structured information, and leverages the XML standard as a means of integrating heterogeneous inter-government data sources containing information of varying structure. We presented the steps needed for information definition, storage, access and maintenance of a system in IAIS. We also described the operation of XSAR, the information access component of IAIS. Finally, we compared IAIS to two other methodologies used for information sharing, and analyzed the pros and cons of each method. XSAR is free software, available under the GNU public license. It can be accessed at http://bajaj.heinz.cmu.edu/XSAR/Xsar/, where users can use it to search XML repositories on the WWW. For future work, we plan on: a) prototyping a tool to facilitate the information definition component of IAIS by allowing different parties to collaborate on producing XML DTDs, and b) evaluating the usefulness of IAIS in real-world case studies of inter-agency information sharing.

---

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Information Definition</th>
<th>Information Storage</th>
<th>Information Access</th>
<th>Information Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Text Method</td>
<td>Easy but structured information cannot be defined</td>
<td>Difficult</td>
<td>Low recall and precision, fast response</td>
<td>Easy</td>
</tr>
<tr>
<td>Database Schema Method</td>
<td>Difficult but structured information can be defined</td>
<td>Easy</td>
<td>High recall and precision, fast response</td>
<td>Difficult. Subsets of data schemas need to be made visible across organizations.</td>
</tr>
<tr>
<td>IAIS</td>
<td>Difficult but structured information can be defined</td>
<td>Difficult</td>
<td>High recall and precision, slow response</td>
<td>Easy</td>
</tr>
</tbody>
</table>

Figure 6: Overall Comparison Results For The Three Methods
APPENDIX 1: ILLUSTRATION OF USAGE OF XSAR

Figure A1-1: Creating Query For Local Test Website

Figure A1-2: Search Result Page for Local Test

Search Results

Your Query

- Start URL: http://trajan.briar.cmu.edu/filons/index.html
- Search Condition:
  1. (LASTNAME = 'Smith') Exact
  2. (FIRSTNAME = 'John') Exact
- Match type: EXACT
- Max Search Time (min.): Unlimited
- Max Link Depth: Unlimited

Search Result

   [See matches] [See whole XML]
   1. Node location: /filons/name

Number of HTML(s) checked: 1
Number of XML(s) checked: 1
Total Time: 0.123 [ms]
Figure A1-3: Result of “See Matches” for local test website

< Felon >
  < Name >
    < LastName >Smith</ LastName >
    < FirstName >John</ FirstName >
  </ Name >
  < SSN >123-789-6555</ SSN >
  < Convictions >
    < ConvictionItem >
      < ConvictionDate >May 9, 1997</ ConvictionDate >
      < ConvictionDescription >Attempt to break and enter a residence</ ConvictionDescription >
      < ConvictionSentence >One Year without parole</ ConvictionSentence >
    </ ConvictionItem >
    < ConvictionItem >
      < ConvictionDate >Oct 1, 1999</ ConvictionDate >
      < ConvictionDescription >Attempted burglary</ ConvictionDescription >
      < ConvictionSentence >Two Years without Parole</ ConvictionSentence >
    </ ConvictionItem >
    < ConvictionItem >
      < ConvictionDate >Oct 12, 2002</ ConvictionDate >
      < ConvictionDescription >Burglary with weapon</ ConvictionDescription >
      < ConvictionSentence >Fifteen Years with Parole Hearing after ten</ ConvictionSentence >
    </ ConvictionItem >
  </ Convictions >
  < ExpirationDate >Jan 1, 2004</ ExpirationDate >
</ Felon >

Figure A1-4: Result of “See whole XML” for local test website

<html version="1.0" encoding="UTF-8" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/1999/xhtml xhtml.xsd">
  <Document>
    <Felons>
      <Felon>
        <Name>
          <LastName>Smith</LastName>
          <FirstName>John</FirstName>
        </Name>
        <SSN>123-789-6555</SSN>
        <Convictions>
          <ConvictionItem>
            <ConvictionDate>May 9, 1997</ConvictionDate>
            <ConvictionDescription>Attempt to break and enter a residence</ConvictionDescription>
            <ConvictionSentence>One Year without parole</ConvictionSentence>
          </ConvictionItem>
          <ConvictionItem>
            <ConvictionDate>Oct 1, 1999</ConvictionDate>
            <ConvictionDescription>Attempted burglary</ConvictionDescription>
            <ConvictionSentence>Two Years without Parole</ConvictionSentence>
          </ConvictionItem>
          <ConvictionItem>
            <ConvictionDate>Oct 12, 2002</ConvictionDate>
            <ConvictionDescription>Burglary with weapon</ConvictionDescription>
            <ConvictionSentence>Fifteen Years with Parole Hearing after ten</ConvictionSentence>
          </ConvictionItem>
        </Convictions>
        <ExpirationDate>Jan 1, 2004</ExpirationDate>
      </Felon>
      <Felon>
        <Name>
          <LastName>Jones</LastName>
          <FirstName>Albert</FirstName>
        </Name>
        <SSN>123-456-7090</SSN>
        <Convictions>
          <ConvictionItem>
            <ConvictionDate>April 17, 1998</ConvictionDate>
            <ConvictionDescription>Driving Under Intoxication</ConvictionDescription>
            <ConvictionSentence>Three Months and Fine of $1000</ConvictionSentence>
          </ConvictionItem>
          <ConvictionItem>
            <ConvictionDate>January 1, 2001</ConvictionDate>
            <ConvictionDescription>Aggravated Assault with a Motor Vehicle</ConvictionDescription>
            <ConvictionSentence>Two Years without Probation and Fine of $20,000</ConvictionSentence>
          </ConvictionItem>
        </Convictions>
        <ExpirationDate>Jan 1, 2003</ExpirationDate>
      </Felon>
    </Felons>
  </Document>
</html>
APPENDIX 2: FLOWCHARTS OF THE HIGH LEVEL LOGIC FOR XSAR

Figure A2-1: Flowchart of the Controller

Figure A2-2: Flowchart of the Model Component

Figure A2-3: Flowcharts of the Threads that Parse HTML and XML Documents
APPENDIX 3: PERFORMANCE RESULTS OF TEST CASES

We present the results of three test cases that illustrate the performance of XSAR.

The first case was a repository created inside the same network as XSAR. Therefore, the target website and the agent server were connected by Ethernet, 10Mbps. The contents of the repository were small HTML files and XML files, less than 4Kbytes each.

The second case was a real XML repository (http://www.xml-cml.org/), which has the conclusive resource for Chemical Markup Language which is used to exchange chemical information and data via the Internet.

The third case was also a real-world case: the Dublin Core Metadata site (http://purl.org/dc/) which is a specification for describing library materials.

Table 1 shows the search time for the cases mentioned above. The search time results clearly depend on exogenous variables like network conditions, PC hardware specifications, and load status. In order to eliminate at least some variation in these factors, we tested each case five times, and present the mean values in Table 1.

Table 1: Search Time

<table>
<thead>
<tr>
<th></th>
<th>Local experimental Environment</th>
<th>CML</th>
<th>Dublin Core</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number / Average size of HTML</strong></td>
<td>16 / 458 [bytes]</td>
<td>1 / 10490 [bytes]</td>
<td>34 / 16966 [bytes]</td>
</tr>
<tr>
<td><strong>Number / Average size of XML</strong></td>
<td>16 / 2756 [bytes]</td>
<td>173 / 10314 [bytes]</td>
<td>27 / 974 [bytes]</td>
</tr>
<tr>
<td><strong>Direct access</strong></td>
<td>2545[ms]</td>
<td>19291[ms]</td>
<td>44637[ms]</td>
</tr>
<tr>
<td><strong>Via cache server</strong></td>
<td>1772[ms]</td>
<td>14577[ms]</td>
<td>40305[ms]</td>
</tr>
<tr>
<td><strong>Ratio of (w/ Cache) / (w/o Cache)</strong></td>
<td>0.696</td>
<td>0.756</td>
<td>0.903</td>
</tr>
</tbody>
</table>
ENDNOTES

1. Xerces is available at xml.apache.org, the GMD-IPSI XQI engine is available at: xml.darmstadt.gmd.de/xql/ and tomcat is available at jakarta.apache.org.

2. While alternatives to DTDs such as XML schemas (www.xml.org) do exist, in this work we focus on XML DTDs since they are the predominant standard for structuring data on the world wide web.

3. We assume a given size of the universe of pages in this section.

REFERENCES


Akhilesh Bajaj is Assistant Professor of Information Systems Management at the H. John Heinz III School of Public Policy and Management, Carnegie Mellon University. He received a B. Tech. in Chemical Engineering from the Indian Insti-
stitute of Technology, Bombay in 1989, an MBA from Cornell University in 1991, and a Ph.D. in MIS (minor in Computer Science) from the University of Arizona in 1997. Dr. Bajaj’s research deals with the construction and testing of tools and methodologies that facilitate the construction of large organizational systems, as well as studying the decision models of the actual consumers of these information systems. He has published articles in several academic journals such as IEEE Transactions on Knowledge and Data Engineering, Information Systems and the Journal of the Association of Information Systems. He is on the editorial board of the Journal of Database Management and the Journal of Electronic Commerce in Organizations. His research has been funded by the department of defense (DOD). He teaches graduate courses on basic and advanced database systems, as well as enterprise wide systems.

Sudha Ram is Eller Professor, Management Information Systems in the College of Business and Public Administration at the University of Arizona. She received a B.S. degree in mathematics, physics and chemistry from the University of Madras in 1979, PGDM from the Indian Institute of Management, Calcutta in 1981 and a Ph.D. from the University of Illinois at Urbana-Champaign, in 1985. Dr. Ram has published articles in such journals as Communications of the ACM, IEEE Expert, IEEE Transactions on Knowledge and Data Engineering, ISR, Information Systems, Information Science, and Management Science. Dr. Ram’s research deals with modeling and analysis of database and knowledge based systems for manufacturing, scientific and business applications. Her research has been funded by IBM, NCR, U.S. ARMY, NIST, NSF, NASA, NIH, and ORD(CIA). Specifically, her research deals with Interoperability among Heterogeneous Database Systems, Semantic Modeling, Data Allocation, Schema and View Integration, Intelligent Agents for Data Management, and Tools for database design. Dr. Ram serves on various editorial boards including Information Systems Research, Journal of Database Management, Information Systems Frontiers, and Journal of Systems and Software. She is also the director of the Advanced Database Research Group based at the University of Arizona.