GIME: A Distributed Geotechnical Information Management and Exchange Architecture Using Web Services

Roger Zimmermann[1], Wei-Shinn Ku[1], Haojun Wang[1], Amir Zand[2], and Jean-Pierre Bardet[2]
[1]Department of Computer Science
[2]Department of Civil and Environmental Engineering
University of Southern California
Los Angeles, CA 90089
{rzimmerm, wku, haojunwa, azand, bardet}@usc.edu

July 14, 2006

Abstract

We present the practical use of Web services applied to a distributed architecture aimed at facilitating the exchange and utilization of geotechnical information. Such data is of critical interest to a large number of municipal, state and federal agencies as well as private enterprises involved with civil infrastructures. The utilization of geotechnical information is currently hampered by a lack of service infrastructure among the heterogeneous data sources operated under different administrative control. We describe a Web services based infrastructure to manage geotechnical data via XML as the common data format.

1 Introduction

Geotechnical information on soil deposits is critical for civil infrastructures. Local, state and federal agencies, universities, and companies need this information for a variety of civil engineering applications, including land usage and development, and mapping of natural hazards such as soil liquefaction and earthquake ground motions. Foremost sources of geotechnical information, geotechnical boreholes are vertical holes drilled in the ground for the purpose of obtaining samples of soil and rock materials and determining the stratigraphy, groundwater conditions and/or engineering soil properties [7]. In spite of rather costly drilling operations, boreholes remain the most popular and economical means to obtain subsurface information. These type of data range from basic borehole logs containing a visual inspection report of soil cuttings to sophisticated composite boreholes combining visual inspection and in-situ, laboratory geotechnical and geophysical tests. Figure 1(a) shows an example transcript of the Standard Penetration Test (SPT), a particular type of geotechnical borehole test. Significant amounts of geotechnical borehole data are generated in the field from engineering projects each year. As data collection technologies improve, more and more geotechnical borehole data from the field and laboratory are directly produced in, or converted to a digital format.

Furthermore, with the recent ubiquity of communication networks – particularly the Internet – the trend towards electronic storage and exchange of geotechnical borehole data has accelerated. One significant constraint is that geotechnical data is collected and managed by a multitude of private and public agencies, such as the U.S. Geological Survey (USGS), the California Department of Transportation (CalTrans), and others. Several pilot efforts are underway that aim to facilitate electronic access to geotechnical information. For instance, the ROSRINE (*Resolution of Site Response Issues from the Northridge Earthquake*) project has produced an integrated system based on a relational database management system (RDBMS), geographic information system (GIS) and Internet Map Server (IMS) to disseminate geotechnical data via the Internet [10]. The USGS is continually publishing seismic Cone Penetration Test (CPT) data through a web-based system managed by the *Earthquake Hazards Program* [6].

The goal of our *Geotechnical Information Management and Exchange* (GIME) project is to overcome the challenges inherent in data sharing among heterogeneous database repositories under different administrative control. Specifically, the following features and goals have guided our design.

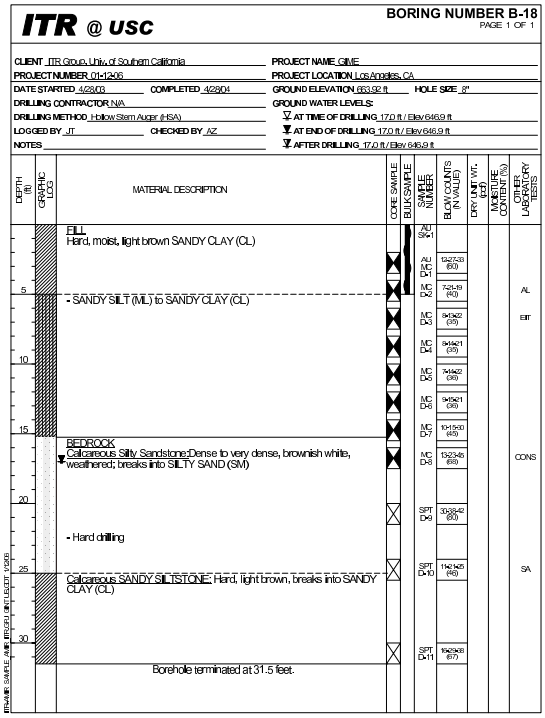

Figure 1(a). Example of boring log.

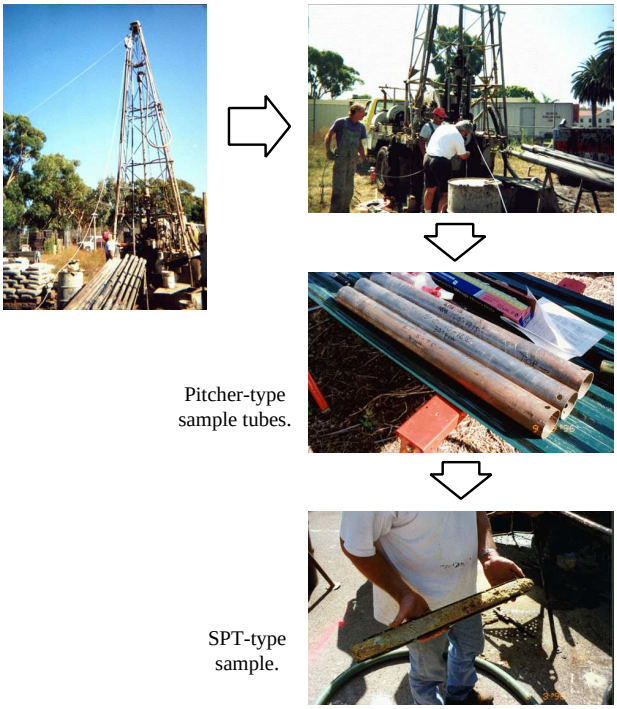

Figure 1(b). Drilling and sampling activities.

Figure 1: The photographs illustrate the geotechnical boring activities from drilling until the soil samples are examined. The result is a boring log showing stratigraphy, physical sampling record and SPT blow counts.

- *Autonomy*: Each of the archives contains data that is maintained by a specific organization (e.g., USGS). For organizational rather than technical reasons, it is undesirable to replicate or cache the data sets at other participating archives. Data sets may geographically overlap.
- *Standardized access*: It is desirable to allow direct, programmatic access to distributed data sets from end user applications. To hide the heterogeneity of the numerous data sources, Web services are employed to provide a standardized interface. Web services build upon the idea of accessing resources (storage space, compute cycles, etc.) from a local machine on a powerful remote computer. Unlike earlier attempts to enable this functionality, Web services are broadly accepted and open standards that are supported by all major industry vendors.
- *Cooperative and efficient query processing*: When presented with a query at any one of the participating database nodes, the overall system must cooperatively execute the request and return all relevant data. For this purpose, an efficient *access method* is required which can rapidly decide which other nodes contain potentially relevant data and which do not. The query must then be forwarded to the candidate nodes and the result returned expediently to the querying host.

We describe our design and implementation of the distributed GIME infrastructure, comprised of a number of geographically distributed spatial databases as illustrated in Figure 2. The rest of this paper is organized as follows. Related work is discussed in Section 2. Section 3 introduces the various components of the GIME architecture and illustrates the usability of the GIME approach with a client application. It also presents the results of our performance evaluation. Conclusions and future research directions are contained in Section 4.

2 Related Work

The Open Geospatial Consortium (OGC, www.opengeospatial.org) has proposed the OGC Web Services (OWS) to facilitate the integration of GIS systems and location services. OWS is designed as a general framework that enables distributed geoprocessing systems to communicate with each other through well-defined interfaces. One of the driving

forces behind the current major revision of this framework, OWS-3, is to provide interoperability between different implementations. It is targeting wide-ranging applications from geospatial sensors to geospatial decision support.

The basic GIME Web services currently use different functions than OWS. While OWS is an excellent framework, we chose a different strategy that is based on the Web services standards from W3C. OWS and the W3C Web services standards evolved somewhat in parallel and they are not currently compatible. While OWS focuses specifically on geospatial applications, the W3C Web services – using the Web Services Description language (WSDL) to describe interfaces and the Simple Objects Access Protocol (SOAP) for data transport – target more general applications. As a result, many of the popular, general development tools and technologies such as Eclipse, .NET, and NetBeans, provide W3C Web service integration. The choices for OWS development are somewhat more limited. One example are the CarbonTools, an open geospatial .NET development toolkit. Because of W3C's wide support, our implementation allows us the flexibility to experiment with the newest techniques such as asynchronous data access (more details on this are described in Section 3.5). Furthermore, it facilitates access to an ample variety and large library of existing Web services, some of which are very useful for an overall application design. For example, Microsoft MapPoint, which provides an extensive set of mapping-related features, can be easily incorporated into GIME applications.

Large bodies of work exists in several related sub-fields such as spatial query processing, distributed query processing, and geospatial Web services. However, there is limited prior work that combines all these topics. Yu *et al.* [12] surveyed various techniques for optimizing queries in distributed databases. They assumed a relational model and queries that are expressed in a QUEL-like tuple relational calculus. A new qualitative spatial relation model and a solution to its consistency problem were proposed by Wang *et al.* [11]. A new method to deduce the constraints of spatial queries is described, thus saving query processing time in distributed GIS systems. Jager *et al.* [8] presented an approach to compose various geospatial Web services through generic visual interfaces and scientific workflow tools. The framework encompassed registering, discovering, composing and executing Web services to support distributed geospatial data processing.

On the commercial side, professional GIS Web service platforms, such as ArcWeb Services from ESRI and the MapPoint Web Service from Microsoft, provide solutions to develop GIS systems. Our example implementation is leveraging open source components; however, proprietary platforms may be attached through middleware wrappers.

3 Geotechnical Information Management and Exchange (GIME)

3.1 Overview

Figure 2 illustrates the architecture of the GIME system. Multiple, distributed geotechnical data archives are accessible via the Internet. Services are provided such that practitioners in the field can directly store newly acquired data in a repository while the data customers are able to access these data sets in a uniform way. By adopting a Web services infrastructure, multiple applications, such as a soil liquefaction analysis or a borehole visualization can be built in a modular fashion.

GIME distinguishes two types of archives on the basis of the data access allowed: read-write (RW) or read-only (RO). RW archives host three geotechnical Web services to provide the interface for distributed applications to store (*File Storage Web service*, FSWS), query and retrieve (*Query & Exchange Web service*, QEWS) and visualize (*Visualization Web service*, VWS) the geotechnical information. RO archives implement only the QEWS Web service. In this case, an on-site database administrator may insert the data directly into the local database. Figure 2 illustrates the components of an RW archive in the upper, left corner.

Finding the relevant data sets required for a specific application among all the geotechnical archives can be a daunting task. To conveniently process the spatial queries and locate the relevant information, we have designed an efficient query routing algorithm for GIME that automatically forwards queries to other known archives and collects the results before returning the data to the application (see Section 3.3). Such forwarding mechanisms can be effective as demonstrated previously by the SkyQuery project [9] and our own distributed query routing techniques [14].

The retrieved geotechnical borehole data is complex and sophisticated in that it contains both well structured and semi-structured elements. In Figure 1(a), for example, the *Material Description* (3^{rd} column from left) field contains free-form text, while some of the other columns are well structured. Therefore, an efficient data format for storage and exchange is required that is suitable for the diversity of geotechnical borehole data. In GIME, we use XML as the preferred container format for both storage and exchange of borehole data [13]. XML offers many advantages over other data formats for borehole data [1, 3]. Its tree-structure and flexible syntax are ideally suited for describing constantly evolving and annotated borehole data. It also lends itself to an automated visualization capability that

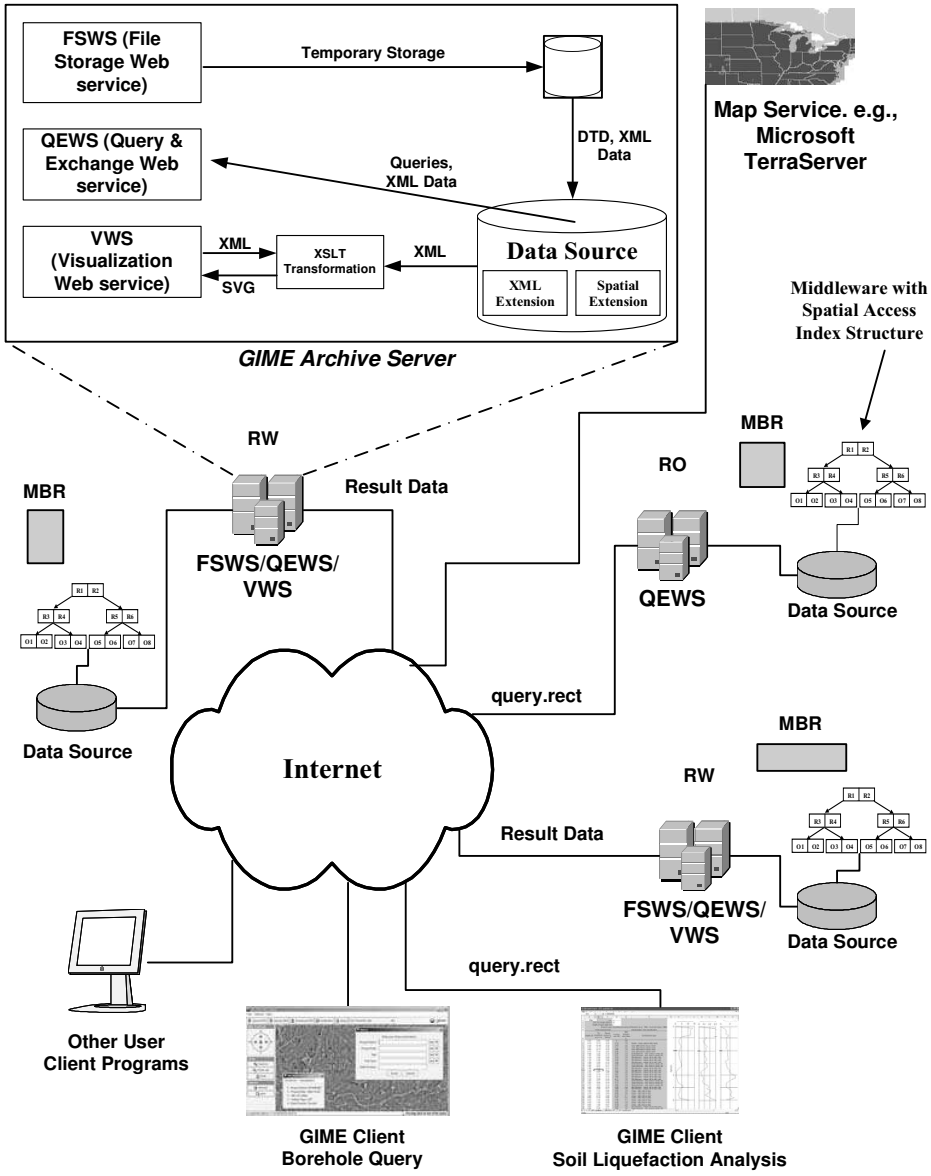

Figure 2: The GIME architecture is composed of multiple, distributed data archives. Some archives are read-only while others allow read and write access. Each archive contains a middleware utilizing replicated spatial index structures.

converts XML geotechnical data into a graphical view similar to the traditional hardcopy format. The output can be presented as *Scalable Vector Graphics* (SVG)[1]. Note that any uploaded data file is first placed in a temporary space where it is validated against the Document Type Definition (DTD) or XML Schema for geotechnical data sets. If the file is accepted the data is then stored in the main database.

3.2 Geotechnical Web Services Functionalities

Web services commonly operate from a combination of a Web and an application server and they can be implemented using many existing tools. Our local GIME prototype testbed utilizes the open source software components Apache Tomcat (web server) and Apache Axis (application server). The application code specific to GIME is embedded within Axis, which is convenient to use: when Tomcat starts, Axis automatically compiles the application codes located in its working directory and generates the necessary object class files.

The three main geotechnical Web services provide a number of specialized methods for programmatic access to

[1] http://www.w3.org/Graphics/SVG/

the geotechnical data. The file storage service provides client programs with an interface to upload their XML data files into the main database. During the upload process, meta-data is extracted and stored. The meta-data includes specific elements of the imported files to facilitate querying. The main purpose of the query and exchange Web service is to facilitate the dissemination of the valuable geotechnical data sets and encourage their usage in broad and novel applications. XML borehole files, although easily readable by computers, become meaningful to geologists and civil engineers only after they are rendered into images (e.g., SVG format). Therefore, generating SVG files is the main purpose of the visualization service. The following is a list of the GIME Application Programming Interface (API) methods.

GeoPutDTD(): Upload and store a new DTD file on the server (FSWS).

GeoGetDTD(): Retrieve the current DTD file (FSWS).

GeoPutXMLFile(): Upload an XML borehole data file and store it on the server (FSWS).

GeoQuery(): Execute a query expression and return a list of unique identification numbers, one for each borehole file in the result set (QEWS).

GeoGetXMLFile(): Retrieve an XML borehole data file based on a unique identification number (QEWS).

GeoVisualization(): Transform the XML borehole file selected with the unique identification number into SVG format on the server (VWS).

GeoGetSVGFile(): Retrieve an SVG borehole file based on a unique identification number (VWS).

3.3 Efficient Query Routing with Spatial Indexing

Given a federation of independently managed spatial database servers, one research challenge is the efficient querying of this distributed infrastructure. Note that the data set in each repository may be disjoint or may overlap with other archives. To avoid that an application must contact each and every repository, we have implemented a distributed query mechanism that efficiently and automatically forwards queries to other known archives and collects the results before returning the data to the application. Repository data sets are spatially indexed at a middleware layer via replicated R-trees or Quadtrees [14]. The concept is illustrated in Figure 2. We first introduce a baseline algorithm for comparison purposes.

Baseline Method: Exhaustive Query Routing Geotechnical data sets are generally large and valuable, and therefore they are professionally managed. We consider this a stable environment where occasionally, but not very frequently, a repository leaves or joins the collective. As a result, we can compile a list of all the participating archives. This list may not be completely up-to-date at a specific time instance, but accurate enough to result in few disruptions. The list is distributed to every archive and a query q that arrives at a specific node is forwarded to all other nodes for exhaustive processing. We call this naive method *exhaustive query routing* (EQR). Even though EQR is inefficient, it is useful as a baseline mechanism to compare our more sophisticated models against.

The metric that we use to compare the different techniques is the total number of messages created in the system to execute a query q and collect the results. A lower number of messages reduces network traffic and indicates better scalability of the system. The number of messages generated by queries with EQR can be represented as $M = 2 \times Q \times (N - 1)$: the overall number of messages M is the product of the total number of queries Q and the number of archives N in the system. The total is doubled because an equal number of result messages are generated.

Query Routing with R-Trees and Quadtrees Spatial Indexing The R-tree [5] and Quadtree [4] families of algorithms are well established for spatial data indexing. Both build a tree-structure that partitions the overall space into successively smaller areas at lower levels of the index hierarchy. R-trees and Quadtrees are very successfully used in the core engines of spatial database systems. We use them in a novel way as index structures across multiple spatial databases to decrease the query forwarding traffic. Specifically, we insert the *minimum bounding rectangle* (MBR) of the data set of each archive into a global R-tree or Quadtree. Because we prefer to avoid a centralized index server we further distribute copies of this global index structure to each archive. During query processing, an archive intersects each query rectangle with the archive MBRs stored in the global index. The query is then only forwarded to candidate archives whose MBR overlaps with the query rectangle, immediately reducing inter-node message traffic significantly. Figure 3 shows an example with a blue, shaded query rectangle intersecting with a green and a brown MBR, respectively. Note that forwarded queries are marked to show that they originated from a server rather than a client to avoid

query loops. The results of forwarded queries are returned to the initially contacted server which aggregates them and returns the set to the client.

The above design requires that the global index structures are synchronized and kept consistent. This necessity introduces overhead in terms of both communication cost and implementation complexity. However, one characteristic of this technique greatly reduces the overhead and makes it an attractive solution. Because the global index structures manage bounding rectangles and not individual data points, changes to the data set of any specific archive only result in index updates if the MBR changes – and this is very infrequent. Consider the following example. An archive manages 1,000 spatial point objects in a two-dimensional space. The MBR is generally defined by four of them: the two points with the most and least x values and the two points with the most and least y values (see examples in Figure 3). Any insertion or deletion of data objects confined within the MBR do not affect the global index; only changes that either stretch or shrink the MBR need to be propagated.

With our spatial indexing mechanism the query traffic is reduced by a factor equal to the *selectivity* S_Q of the query. The selectivity, $0 \leq S_Q \leq 1$, estimates what fraction of the total number of archives may need to be contacted. Similarly, the update message traffic U is diminished by the factor $0 \leq S_U \leq 1$ that describes how many of the data updates (including insertions and deletions) actually result in global index changes. Consequently, the aggregate number of messages is affected by the frequency of data updates in the system. However, we have found in our experiments that the update traffic is a negligible fraction – less than 5% – of the overall message traffic. As described in the next paragraph, global indexing generally results in a significant reduction of message traffic compared with EQR, with the benefits being highest when few updates must be propagated.

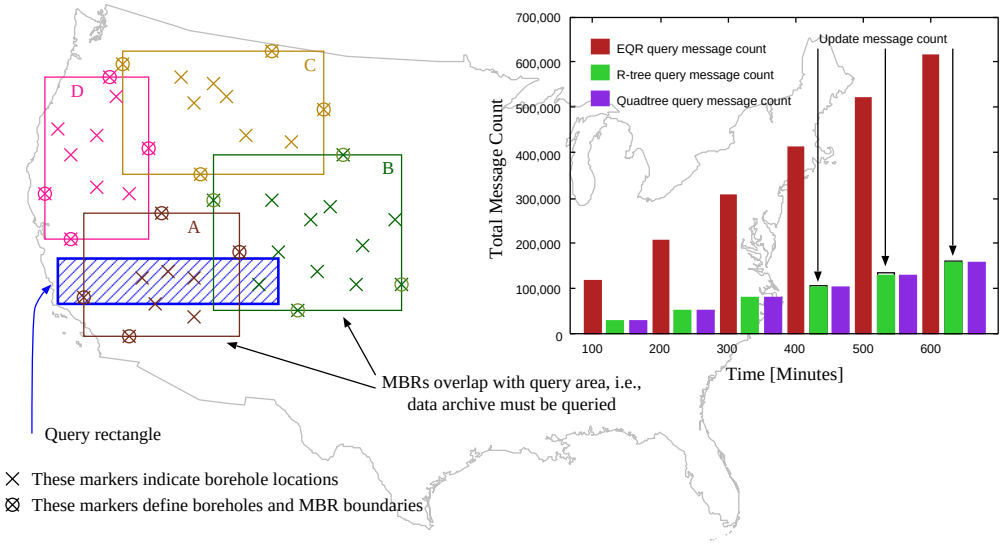

Figure 3: Example of query routing. The blue query rectangle intersects with two MBRs (brown A and green B) which results in those archives being queried for data. The message traffic is greatly reduced (shown on the right) when only relevant archives are contacted.

Experimental Validation To validate the efficiency of our query routing design with a controlled and large quantity of access traffic we ported the GIME modules to a simulation environment. The query routing component was implemented with two plug-in modules to enable either the R-tree and the Quadtree algorithms. In a distributed environment, the search complexity is dominated by the communication overhead between servers. Therefore, the goal of our simulation was to quantify the query routing traffic generated by processing a sequence of spatial range queries and updates. If a query window intersected with several server MBRs, then the query was forwarded to each repository. Tree update information was broadcast to all servers. We performed our experiments with both synthetic and real-world spatial data sets and we also created a simulation module to analyze the message traffic generated by the exhaustive query routing (EQR) mechanism with the same event sequence. Consequently, we were able to measure the performance differences between the two approaches.

Figure 3 illustrates that our design improves the query routing performance significantly. The tree-based designs result in a decrease of approximately 60% to 70% of inter-server message traffic compared with exhaustive query

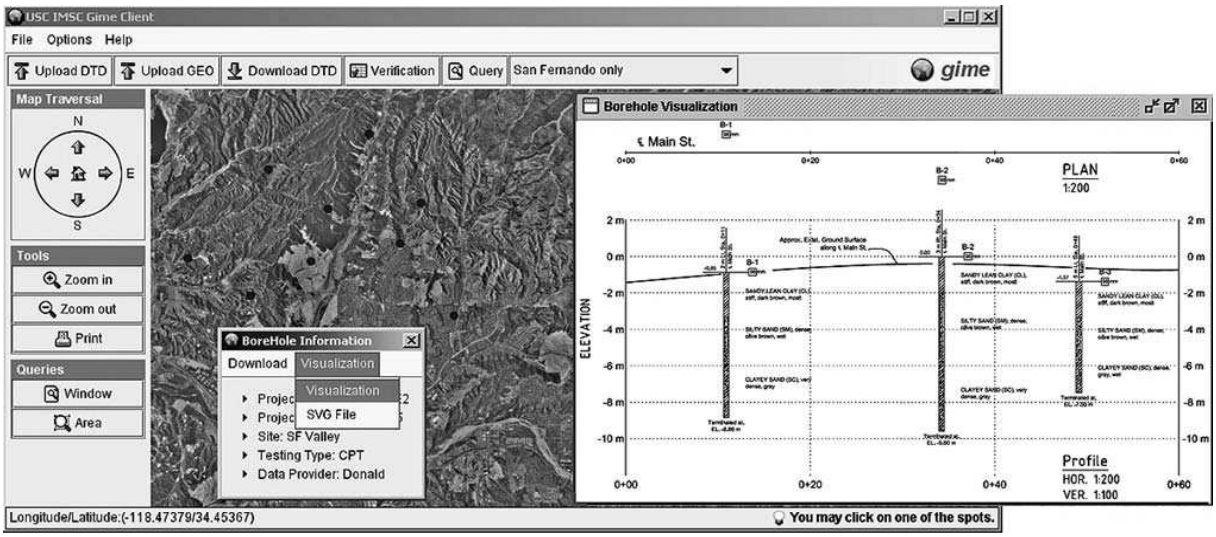

Figure 4: A sample Borehole Query & Drafting client application. After a query has been issued a small pop-up display (the left window) illustrates the metadata of a borehole when the user clicks on one of the dots (indicating a borehole location) on the map. The fence diagram rendering (the right window) is generated from three selected XML data files.

routing (with query window sizes of 10% to 20% of the data area). These results indicate that our techniques are expected to scale well.

We gained some additional insights into our techniques. For the R-tree based design, every MBR change translates into an index structure update. By contrast, we found that in the Quadtree design the index structure updates are reduced by an order of magnitude. Hence we can conclude that the update message traffic to synchronize distributed Quadtrees is much lower than for R-trees. This is because most MBR updates do not affect the Quadtree structure, hence no index synchronization is necessary.

3.4 GIME Client Application Example

The feasibility of the GIME concepts is illustrated with a sample client application shown in Figure 4. This borehole query and drafting (BQ&D) application is written in Java and demonstrates all the features of GIME, i.e., query and visualization of borehole data, and exchange of XML files. The background map is assembled from aerial images retrieved from the Microsoft TerraServer Web service [2]. A query window can be selected graphically and the meta-data of matching borehole files are obtained from GIME and their locations displayed as dots over the background map (Figure 4). Borehole meta-data can be viewed by clicking on the dots.

One advantage of a Web services infrastructure over traditional browser based applications is that raw data can be programmatically accessed and locally processed. For example, stand-alone engineering programs can import information from GIME. This integration is illustrated with a drafting capability in our client. The drafting component produces fence diagrams and Log of Test Borings (LOTBs) along user defined alignments, based on the data retrieved from GIME Web services. A fence diagram is a two dimensional interpretation of the soil stratigraphy along a (usually) vertical plane. The right-most pop-up window in Figure 4 illustrates the concept of a fence diagram. Fence diagrams and LOTBs are the fundamental tools for civil engineers to perform major geotechnical studies. Traditionally, the production of these diagrams has required significant drafting efforts. The drafting application reduces this effort by automating the drafting process and directly accessing the required borehole data via the GIME infrastructure.

An earlier version of the visualization component generates SVG files using Apache Batik. It can be downloaded from the GIME website and installed as a standalone Java program along with the Batik Squiggle package to display SVG files. More details are available from the GIME website at http://datalab.usc.edu/gime/.

3.5 Asynchronous Data Access

Even though the GIME infrastructure provides an efficient spatial query routing algorithm, accessing and moving data in widely distributed environments naturally introduces communications overhead and delays between servers. Providing low system latency and good response time can be especially challenging with services that move large volumes of data, for example retrieving map information from a Web service such as TerraServer. Web services provide a synchronous access method and in the worst case, a service might become a bottleneck in the system. Hence a system designer needs to carefully consider performance issues when handling large data sets via Web services.

Novel techniques for asynchronous data access, such as Asynchronous JavaScript and XML (AJAX), aim to reduce the latency of system responses and increase interactivity. The implementation of AJAX is based on the observation that for a number of tasks only small amounts of data need to be retrieved from the server incrementally. In AJAX, a client adaptively retrieves data from a server based on – for example – a user's interaction and navigation. As a result the user experiences prompt system responses. While AJAX is geared towards interactive applications, the technique can also be useful for large-scale simulations that progressively access volumes of data.

We have implemented a prototype of the GIME system supporting asynchronous data access. On the client side, the map data is retrieved from Yahoo! Maps Web Services via AJAX so that the latency of map retrieval is diminished. In addition, we defined the area of the map tiles displayed on the client program as the borehole data access area. Once the user submits a spatial query, the client program only retrieves the query results that overlap with the visible map area. If the user changes the displayed area of map tiles and navigates to a different location, the client program will calculate the data access area and retrieve the query result set correspondingly. In essence, the data retrieval function of GIME is adaptively invoked based on the user's interactions. Initial user experience has confirmed that the responsiveness of the system is notably enhanced.

4 Conclusions and Future Research Directions

We presented our Geotechnical Information Management and Exchange architecture aimed at facilitating the utilization of geotechnical information. We illustrated the usefulness of this design with a relevant application that implements direct programmatic access to geotechnical data via our Web services. The Web services are being presently tested in realistic work environments in collaboration with municipal, state, and federal agencies, as well as international partners. We plan to extend our work in multiple directions. First, we anticipate that additional Web services will be necessary once more sophisticated applications will be deployed. Additionally, for some applications the progressive streaming of large query result sets would be beneficial, such that asynchronous processing of – for example – time-consuming simulations can commence as quickly as possible.

Acknowledgements This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), CMS-0219463 (ITR), IIS-0534761 and equipment gifts from the Intel Corporation, Hewlett-Packard, Sun Microsystems and Raptor Networks Technology.

References

[1] AGS. *Electronic Transfer of Geotechnical and Geoenvironmental Data*. Association of Geotechnical and Geoenvironmental Specialists, 3rd Edition, United Kingdom, 1999.

[2] T. Barclay, J. Gray, E. Strand, S. Ekblad, and J. Richter. TerraService.NET: An Introduction to Web Services. Technical Report MSR-TR-2002-53, Microsoft Research, June 2002.

[3] COSMOS/PEER-LL 2L02, USC. *User Survey. Archiving and Web Dissemination of Geotechnical Data Website*, 2002. URL http://geoinfo.usc.edu/gvdc/.

[4] R. Finkel and J. Bentley. Quadtree: A Data Structure for Retrieval on Composite Keys. *ACTA Informatica*, 4(1):1–9, 1974.

[5] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, Massachusetts, June 18-21, 1984.

[6] T. Holtzer. Distribution of USGS CPT Data via the Web. In *Procedings of the COSMOS/PEER-LL Workshop on Archiving and Web Dissemination of Geotechnical Data*, Richmond, CA, October 2001.

[7] R. Hunt. *Geotechnical Engineering Investigation Manual*. McGraw-Hill, New York, 1984.

[8] E. Jaeger, I. Altintas, J. Zhang, B. Ludäscher, D. Pennington, and W. Michener. A Scientific Workflow Approach to Distributed Geospatial Data Processing using Web Services. In *SSDBM*, pages 87–90, 2005.

[9] T. Malik, A. S. Szalay, T. Budavari, and A. R. Thakar. SkyQuery: A Web Service Approach to Federate Databases. In *Proceedings of the First Biennial Conferenc on Innovative Data Systems Research (CIDR 2003)*, pages 188–196, Asilomar, CA, January 5-8 2003.

[10] J. Swift, J.-P. Bardet, J. Hu, and R. Nigbor. An Integrated RDBMS-GIS-IMS System for Dissemination of Information in Geotechnical Earthquake Engineering. *Computers & Geosciences*, 2002. Accepted for publication.

[11] S.-s. Wang and D.-y. Liu. Spatial Query Preprocessing in Distributed GIS. In *International Conference on Grid and Cooperative Computing*, pages 737–744, 2004.

[12] C. T. Yu and C. C. Chang. Distributed Query Processing. *ACM Computing Survey*, 16(4):399–433, 1984.

[13] R. Zimmermann, J.-P. Bardet, W.-S. Ku, J. Hu, and J. Swift. Design of a Geotechnical Information Architecture Using Web Services. In *Proceedings of the Seventh World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2003)*, Orlando, Florida, July 27–30, 2003.

[14] R. Zimmermann, W.-S. Ku, and W.-C. Chu. Efficient Query Routing in Distributed Spatial Databases. In *Proceedings the 12^{th} ACM International Symposium on Geographic Information Systems, ACM-GIS 2004, November 12-13, 2004, Washington, DC, USA*, pages 176–183, 2004.