ICA UPIMap2004, Tokyo
<C-02>

Mapping on the Grid: From Spatial Web Services to Mobile Clients

Ilya Zaslavsky, Ashraf Memon, Pavel Velikhov, Chaitanya Baru
San Diego Supercomputer Center, UCSD, 9500 Gilman Dr., La Jolla, CA 92093-0505

{zaslavsk|amemon|pvelikhov|baru}sdsc.edu

Abstract. *This paper combines data grid and web services techniques with information mediation and online mapping approaches to outline a scalable infrastructure for ubiquitous mapping where heterogeneous spatial data can be discovered, accessed, queried, integrated and visualized as maps on both desktop and mobile devices. The grid services infrastructure elements we describe, are being developed within GEON (the Geosciences Network) project. The focus of the paper is services that support ubiquitous mapping: from federating multiple spatial data sources to assembling composite maps from heterogeneous map fragments on user queries, and to delivering vector map content to various devices. We present system-independent abstractions of integrated views used by GEON mediation services for generating composite maps, tracing them from XQuery specification to implementation using grid services.*

Keywords: *web services, grid computing, information mediation, SVG, mobile devices*

1. Introduction

Ubiquitous cartography implies a common set of abstractions and interfaces for spatial data that make them accessible from a variety of client devices, to support on demand generation of maps tuned to particular locational or thematic context. Important advances in standardizing interfaces to spatial data sources have been made in recent years, through the efforts of Open GIS consortium (OGC) and other standards bodies, as well as with proliferation of industry (such as ESRI's ArcIMS, see ESRI 2004) or community (such as Minnesota MapServer)-supported internet map servers easily decoupled from client-side mapping applications.

However, ubiquitous mapping requires an additional degree of flexibility in the system, where distributed sources of map data can be seamlessly engaged in mapping regardless of the particular data formats and models they support, differences in language and semantics, spatial coverage, etc. For example, visiting San Diego county a user may get more detailed and up-to-date maps requesting them from map services independently maintained by San Diego Association of Governments (SANDAG), San Diego Geographic Information Source (SanGIS), City of Chula Vista, etc., rather than connecting with a predefined set of services available at the federal level. These services might have different schemas, support different types of access, or generate different types of map fragments. Furthermore, a map requested by the user, may have to be assembled as a union of these heterogeneous fragments, as no single source may cover the area of interest or provide sufficient data. Finally, the maps assembled "on the fly" from one or more discovered data sources, may be requested from different client devices, with different bandwidth, document size, rendering speed and client-side interactivity expectations. For example, streaming map features from an ArcIMS feature service may be appropriate for a desktop applications while bringing a mobile phone map viewer to its knees; the mainstream map serving technology (i.e. each map update involving a client-server round-trip and new image generation at the server) may not be appropriate for a navigation application on a mobile device, etc. Yet another requirement for a ubiquitous mapping system is a higher level of fault tolerance as there is little central control over independently managed spatial data servers.

The goal of this paper is to outline a scalable and flexible mapping infrastructure that can address these challenges. In particular, the infrastructure requirements suggest a need for an extensive middle tier to handle intelligent registration and discovery of map sources, and assembling maps on user requests from selected sources. We will base our discussion on the several critical middleware elements which are being developed in the course of the Geosciences Network (GEON) project. GEON is an NSF-supported large ITR (Information Technology Research) project focused on the development of cyberinfrastructure for geosciences research, through collaboration between geoscientists and computer scientists [GEON 2003]. We extend the GEON infrastructure by adding services specifically designed to support mapping on various devices, including mobile devices. This requires development of system-independent abstractions for map assembly, and tracing how they can be translated into particular implementations of grid services for ubiquitous mapping.

The paper is structured as follows. Section 2 describes grid services infrastructure being developed within GEON, focusing on the common exchange protocol, scalability and flexibility. The core of the discussion here is GEMS (Grid-Enabled Mediation Services), a collection of services that support integration of heterogeneous grid data resources. In the following section, we discuss mapping aspects of mediation, in particular the specification of declarative integrated views over distributed spatial data sources. Based on this specification, we outline two general purpose implementations of map assembly. A description of map assembling grid services implemented as a mediator-level ArcIMS service is given in section four. It is continued in Section 5 with a detailed description of a web service for generating and optimizing SVG (Scalable Vector Graphics) maps for mobile devices. The paper is concluded with a summary and research and development outlook.

2. The GEON Data Grid

2.1 The Architectural Outline

Data federation environments referred to as data grids, have been recently proposed as a strategy for seamless standards-based management and integration of heterogeneous distributed data resources [Foster et al. 2001, 2002]. Grid services represent language- and system-independent re-usable functional components that follow standardized description templates (WSDL – see W3C 2003a), can be invoked using standard protocols (eg, SOAP – see W3C 2003b), and follow standard

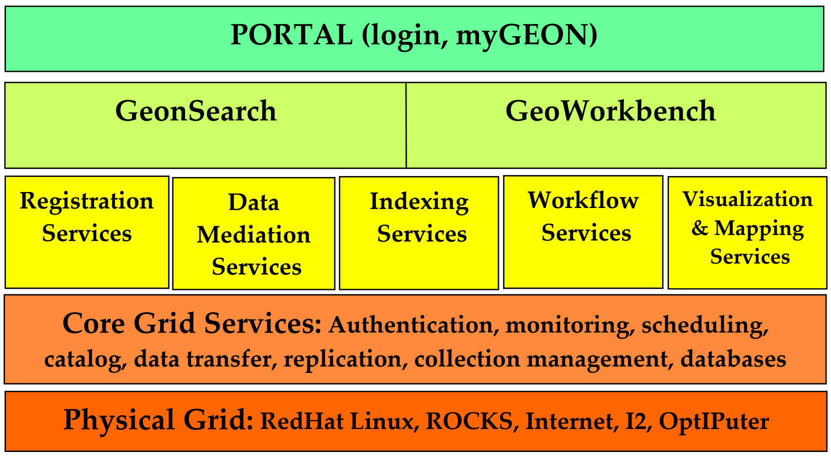
PORTAL (login, myGEON)
GeonSearch
GeoWorkbench
Registration Services
Data Mediation Services
Indexing Services
Workflow Services
Visualization & Mapping Services
Core Grid Services: Authentication, monitoring, scheduling, catalog, data transfer, replication, collection management, databases
Physical Grid: RedHat Linux, ROCKS, Internet, I2, OptIPuter

Figure 1. The GEON services infrastructure (Source: GEON poster, 2004)

mechanisms for service security, persistent state and lifetime management, data access and integration, authentication, etc. They are outlined in Open Grid Services Architecture (OGSA), standardized through the activities of the Global Grid Forum [GGF 2003], and explored in the GLOBUS project [GLOBUS 2003].

As an example, an architectural outline of GEON (The Geosciences Network) data grid is shown in Figure 1. GEON develops grid services-based cyberinfrastructure to support large-scale collaborative research and data integration in the geosciences. From the GEON portal (http://www.geongrid.org) users can access GeonSearch and GeonWorkbench environments, which expose thematically linked application services, including *registration, indexing, mediation, workflow, and visualization services*. These application services rely on a layer of core grid services which support data movement between GEON grid nodes, user access control and security, load balancing and network weather service, and communication with the underlying physical grid infrastructure.

2.2 Grid-Enabled Mediation Services (GEMS)

Information mediation is central to the GEON grid. It enables users to execute complex queries against multiple data sources, and assemble heterogeneous query results into composite presentations as maps, charts and tables. SDSC Grid Enabled Mediation Services (GEMS) follow the common 3-tier wrapper-mediator architecture [Wiederhold 1992]. According to this model, a middleware software component called the *mediator* parses and rewrites a user query generating a series of query fragments against individual data sources, orchestrates query execution, and composes query results into a virtual document returned to the user. In addition to the mediator, middleware services include a collection of query rewriting services (ontology-based and data quality-based – see Lin and Ludäscher 2003, Manpuria et al., 2003, Zaslavsky et al., 2003), and services for spatial results assembly. Within GEON, these services, as well as other components of the system, represent loosely-coupled grid services that are independently registered, updated and invoked. A mediator-level registry of data sources and services contains, for both GEON-hosted and other sources, source schemas and capabilities, as well as information about source placement within spatial, temporal and conceptual (ontology) contexts. Based on this registry, *declarative integrated views* over distributed sources are specified in XQuery, a standardized query language for XML [W3C 2004a]. More details about mediation and grid services in GEMS are given in [Zaslavsky et al., 2003].

Spatial information sources in GEON include grid data nodes based on PostgreSQL, Oracle and DB2 with spatial options, ArcIMS and OGC's WMS (OGC 2000) servers, as well as shapefile collections and pure XML/GML (Geography Markup Language – see OGC 2001) sources. GEON-hosted spatial sources are wrapped in *grid wrappers* that expose source schemas as GML, and present a comparable set of methods dependent on the underlying GIS. For example, WMS sources are accessed via the standard GetMap, GetCapabilities and GetFeatureInfo requests, while ArcIMS sources expose similar methods that are implemented using ArcXML's GET_SERVICE_INFO, GET_FEATURE_COUNT, GET_IMAGE, GET_FEATURES and GET_EXTRACT requests. The source wrappers accept a SOAP message from the mediator, evaluate the validity of a grid security certificate in the SOAP header, and – if authorized - convert the content of the request into native source queries (for example, into ArcXML requests for ArcIMS servers), execute them, and wrap outgoing query results in a SOAP envelope. Such grid service-based organization provides several benefits for handling heterogeneity in a distributed system. The integration problem is reduced to formulating queries against a homogeneous data model (GML) with a common set of operations (though sources expose different composition of operations depending on the underlying server implementation). All components are system-independent, and can be discovered and invoked in a similar fashion. Additionally, higher level of fault tolerance is achieved due to loose coupling between resources, metadata and data replication mechanisms, and adherence to a common interchange standard, when a failure in a particular service component won't bring down the entire system.

To make the mediation system support spatial data integration and ubiquitous mapping, we need to specify declarative *spatial integrated views*, expressed in a compatible and system-independent fashion, and implement services for processing such views. These integrated views should be meaningful from cartographic perspective, i.e. provide for logical ordering of layers and accommodate abstractions for various standard map components. The XQuery-based specification of spatial integrated views is discussed in the following section, while two implementation examples follow in the subsequent sections.

```
Declare Function s-view
($name as xs:string, $coordsys as xs:int,
$minx as xs:double, $miny as xs:double,
$maxx as xs:double, $maxy as xs:double
$age as xs:string, $distance_r as xs:double, $distance_g as xs:double) As element() {

let $env := envelope($minx,$miny,$maxx,$maxy) cast as ogc:polygon
return
<Sview>
   <name> {$name }</name>
   <projection>{ $coordsys }</projection>
   <envelope>$minx, $miny, $maxx, $maxy</envelope>
   <layers>
   <mview>
   {
       for $ocean in source("basemap")//ocean
       where overlap(projection($ocean/Shape,$coordsys), $env ) = 1
       return
       <source>{ $ocean }</source>
   }
   </mview>
   <mview>
   {
       for $land in source("basemap")//land
       where overlap (projection($land/Shape,$coordsys), $env) = 1
       return
         <source>{ $land }</source>
   }
   </mview>
   <mview>
   {
       for $hydrology in source("usgs_hydro")//river
       where overlap (projection($hydrology/Shape,$coordsys), $env) = 1
       return
       <source>{ $hydrology }</source>
   }
   </mview>
   <mview>
   {
       for $rock in source("geon_rocks")//rock
       where overlap (projection($rock/Shape,$coordsys), $env) = 1
       return
       <source>{ $rock }</source>
    }
   </mview>
   <mview>
   {
       for $fault_rock in faults_within_distance_of_rocks_of_given_age($age,
$distance_g)
       where overlap (projection($fault_rock/Shape,$coordsys), $env) = 1
       return
       <source>{ $fault_rock }</source>
   }
   </mview>
 </layers>
 <mviews>
   <mview>
   {
       for $fault_ river in faults_near_rivers($distance_r)
       where overlap(projection($fault_river/Shape,$coordsys), $env) = 1
       return
       <source>{ $fault_ river }</source>
    }
    </mview>
</m_views>
</Sview>
```

Figure 2. An *S-view* specification for a sample map

3. Spatial Mediation on the Grid

3.1 Declarative Integrated Views Specification

To federate over available spatial data and computational resources, we introduce two related integrated view abstractions: spatial integrated views (called *S-views* below), and mediator integrated views (*M-views*). An *S-view* represents a collection of distributed mapping services and views, which together form a map with a given initial spatial extent, projection, units, etc. *S-views* reference one or more *M-views* that describe map layers as well as valid queries against multiple services within the *S-view*. The output of the XQuery is an XML fragment which represents a composite map configuration document specifying a sequence of spatial layers to be shown on the map in a given projection and spatial extent, and a collection of queries to be exposed to the user as part of map interface. A sample XQuery specification of an *S-View* is shown in Figure 2, while Figures 3-4 provide examples of *M-views* referenced from the *S-view.*

The *M-views* can simply reference a single data service, or integrate over several of them, based on spatial or attribute joins. For example, Figure 3 shows a parameterized query "find all faults within a user-specified distance from geologic formations of a given age", which is included as a thematic layer in the map. (Note the "*ontology"part-of*" operation that wraps the "age" parameter. This construct handles geologic age hierarchy stored as an OWL (Web Ontology Language – W3C 2004b) file associated with the service, and expands a given "geologic age" to include its child concepts: for example, querying "Quaternary" geology will also include "Pleistocene", "Miocene", "Holocene", etc. Handling spatial and attribute ontologies is an important part of GEON [Lin and Ludäscher 2003], and is useful for ubiquitous mapping due to the need of defining views over multiple resources with potentially different semantics, but it is beyond the scope of this paper.) Alternately, an *M-view* may be translated not into a map layer but into a parameterized query exposed on the client interface (Figure 4). Once *M-views* are built, they can be queried as all other data sources, and selection queries against them are no more difficult to formulate than queries against primary sources.

```
Declare Function faults_within_distance_of_rocks_of_given_age
($age as xs:string, $distance as xs:double) as element() {
        for
                $fault IN source("faults")//fault,
                $rock IN source("geon_rocks")//rock
        where
                overlap(buffer($rock/Shape, $distance), $fault/Shape) = 1 AND
                Ontology:part-of($rock/age, $age)
        return $fault
}
```

Figure 3. A sample M-view declaring a layer of faults within a given distance from geologic formations of a given age.

```
Declare Function faults_near_rivers
($distance as xs:double) AS element() {
        for
                $fault IN source("faults")//fault,
                $river IN source("hydrology")//river
        where
                overlap(buffer($river/Shape, $distance), $fault/Shape) = 1
        return $fault
}
```

Figure 4.A sample M-view declaring a parameterized query "find faults within a given distance from rivers".

Given an *S-view*, a mediator-level map assembly service constructs a composite map from the referenced data services (generating an ArcIMS-based grid service instance if necessary, or constructing an SVG document), and presents both the map and the available *M-views* at the mapping portal. In a typical session, user loads an *S-view*, and queries the collection of spatial sources defined in the view via one or several forms built over the parameterized *M-views*. The *M-views* are then processed by the mediator, which parses the query, generates query fragments against individual services and relays them as web service requests to source wrappers. The results either update the map (if the *M-view* belongs to the <layers> group), or are returned in a non-map form, as tables or charts.

Various implementations of integrated views in mediators (*M-views*, in our case) have been discussed in the literature, including studies focused on spatial information integration [e.g., Gupta et al, 1999, Shimada and Fukui 1999, Boucelma et al. 2002], so we focus below on implementing *S-views*, which have not been systematically considered. Once generated by the mediator in response to a user query, *S-views* need to be processed into composite maps in a way dependent on the type and capabilities of the requesting output device, available rendering mechanism, network bandwidth, etc. Previously, we have described several special cases of cartographically-conscious map assembly, where data integration was simplified by relative homogeneity of map fragments returned by each service [Zaslavsky et al. 2003]. Here, we are interested in generic grid-based mechanisms for composite map assembly given mediator-generated *S-views*.

4. Implementation 1: ArcIMS-based Map Assembly Service

The purpose of the spatial results assembly service is to generate a composite map from heterogeneous map fragments retrieved from individual data sources, in a way consistent with cartographic design principles. This implies establishing a logical sequence of layers of different types, adhering to a consistent and meaningful coloring scheme across different sources, including common map elements such as labels, title, scale bar, etc.

The task is relatively simple if each source returns result fragments as images, and these images have a common spatial extent and consistent symbolization. In a general case, when query result fragments are of arbitrary types, including raster images, [compressed] shapefiles, GML, ArcXML with coordinate information for each feature, etc., map assembly is performed by a special *map assembly service*. The map assembly service gets invoked after the user formulates an XQuery request against distributed spatial data services (step 1 in Figure 5), and the query is executed by the mediator which issues grid service calls against each source (step 2), The assembly service then:

- receives handles to documents generated by each source (step 3);
- converts the sequence of layers and the spatial extent information expressed in *S-view*, into a map configuration document (step 4);
- selects one of several map assembly templates stored in *command.xml,* which bind together available processing components (for file transfer, data conversion, uncompression, etc.) into a map assembly workflow (step 5);
- retrieves result fragments generated by the sources in response to each *M-view* query, into a local staging area, and transforms them as specified in the previous step. This step (step 6 in Figure 5) involves *File Transfer Service* (file transfer over HTTP or via GridFTP web service), *Uncompress* Service (since result fragments are typically shipped in compressed format for efficiency), and *Data Conversion Service* (responsible for converting result fragments into formats that ArcIMS can import);
- rewrites the current map configuration document into a valid ArcXML configuration file (i.e. referencing the local result fragments), and generates a new ArcIMS image service based on the configuration (inside the *Image Assembly Service* - step 7).

A mapping client then interacts with this dynamically-generated service, ideally without re-querying individual sources on each user request. To manage service creation and destruction in an explicit standard fashion, we implemented it using OGSA grid service *Factory* interface, and *SoftStateDestruction* and *ExplicitDestruction* interfaces for lifetime management. The *Factory* interface creates a new grid service instance and returns a *Grid Service Handle* (GSH), which in turn can be used to retrieve the service's WSDL description from the *Grid Service Reference* (GSR) for subsequent querying. The service must be destroyed when additional user requests exceed the capabilities of the service (say, if the user zooms out beyond the area covered by the transient ArcIMS service), or after certain period of inactivity.

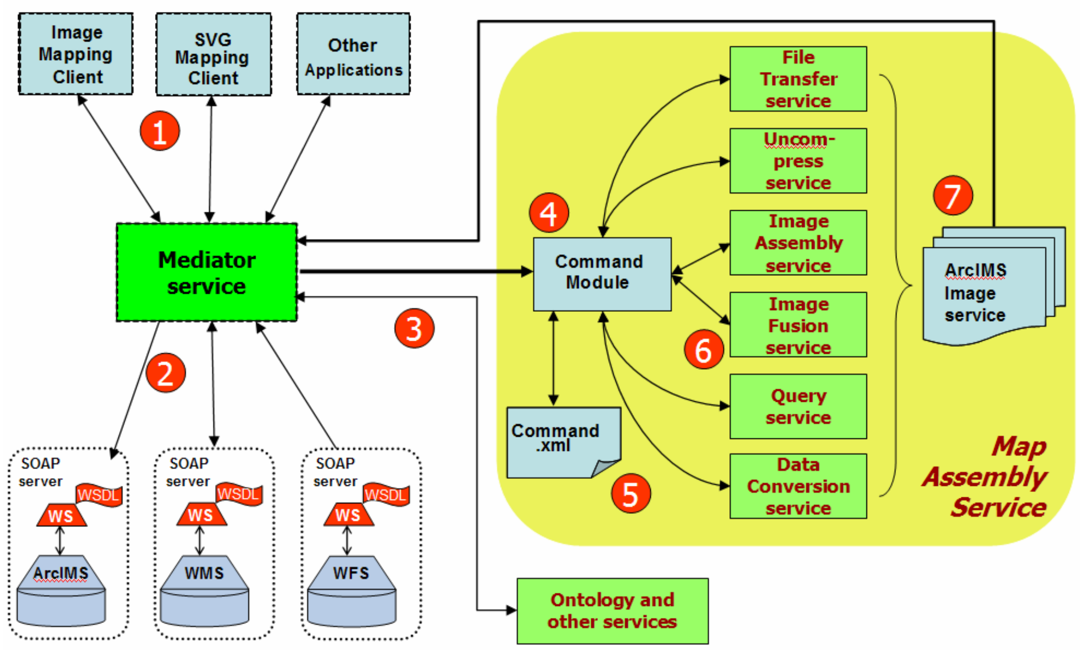

Figure 5. Internal organization of the map assembly services within a spatial wrapper-mediator system.

5. Implementation 2: SVG Generator Service for Mapping on Mobile Devices

SVG provides an attractive platform for mobile mapping applications that require continuous map updates, higher level of interactivity compared to the mainstream internet mapping technology, and integration of spatial information from multiple sources. Converting a mediator-generated *S-view* into an SVG document fragment is another way to implement a map assembly service. The organization of the SVG-based map assembly service generally repeats that of ArcIMS-based map assembly. Based on the map configuration document generated by the mediator as an S-view, the *SVG Generator service*:

- initializes an appropriate SVG map template (*set_map* method), which includes initially empty SVG groups for area, line, point and cosmetic groups of layers and sets SVG canvas coordinates (as implemented in AxioMap which was used as the prototype of the services [Zaslavsky 2000, Zaslavsky et al., 2002]);
- issues SOAP requests against spatial sources in the S-view, to retrieve capabilities of each source, including rendering information for each layer requested from the service (*get_capabilities* method),
- inserts rendering information obtained in the previous step, into the <def> group in SVG, so that rendering styles can be referenced from each layer group (*add_renderer* method)
- requests coordinate information for each layer specified in the *S-view*, and adds it to appropriate groups in the SVG template (*add_point_layer*, *add_line_layer*, *add_polygon_layer, add_labels* methods). The requests against ArcIMS sources are generated as ArcXML GET_FEATURES requests, and the responses are transformed into groups of SVG *path*, *text*, *rect* or *circle* elements.

Depending on the requesting client, the SVG document may have different size and content, and support different kinds of user interaction. For example, creating an SVG Tiny [W3C 2003c] document for rendering on a mobile phone, the service applies a series of optimizations to reduce the size of the document and improve rendering speed, at the expense of supporting feature identity requests. Alternately, for a full SVG-compliant renderer (such as Adobe's SVG 3.0 browser plugin, www.adobe.com/svg) the SVG Generator service scales the SVG coordinate space differently, and adds a variety of user interface controls and event listeners to the SVG-Tiny core.

The procedures for optimizing coordinate geometry implemented in the SVG generator service include:

1. *Adjusting to screen size of target device:* reduces coordinates to 3-digit integers for small screen devices such as PDAs and mobile phones, and 4-5-digit integers for desktop clients. In our experiments, this simple operation alone reduces the size of the SVG document by 50-60%, with minimal increase in processing time.
2. *Collapsing individual feature descriptions into a smaller number of large paths.* SVG rendering engines are known to be more efficient when they process a smaller number of objects. If the client environment does not support identify requests on geometric elements (or doesn't support mouse events in general, like the canonical SVG Tiny), then sacrificing identity of individual features by concatenating them into a single path doesn't worsen the output. On average, this leads to further 10-20% decrease in SVG file size.
3. *Concatenating path fragments.* This optimization corrects digitization artifacts (in street data in particular) when, within a single path definition, there are multiple coordinate chains which may be concatenated into a smaller number of chains. This procedure, therefore, does not sacrifice geometric accuracy or identity of features in order to improve rendering speed. For a typical publicly available street layer originating from TIGER files, we experienced a 5-7% decrease in the size of resultant SVG documents.
4. *Line generalization.* Applying Douglas-Poiker line simplification procedure [Douglas and Poiker 1973] further improves performance of the SVG generator.

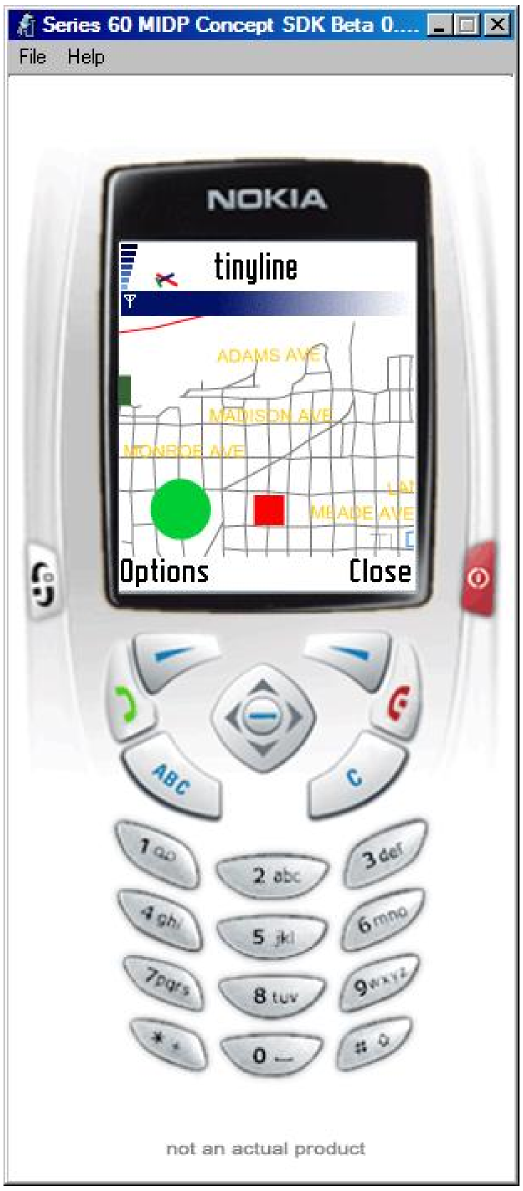

Figure 6. An SVG document fragment for a portion of San Diego displayed in TinyLine SVG viewer.

Overall, these optimizations led to more than 3 times reduction in the size of our sample SVG Tiny documents, without any visible degeneration in geometry or significant increase in processing time. A sample output of the SVG generator, for a fragment of San Diego, is shown in Figure 6. The SVG file is rendered in a mobile device using the freely available TinyLine SVG toolkit for J2ME devices [Girow 2002]. This ability to construct vector maps on J2ME-enabled mobile devices from widely available ArcIMS sources has several important advantages over other emerging mapping solutions for mobile phones, including: reliance on existing infrastructure of spatial data servers instead of a dedicated proprietary non-standard infrastructure, ability to discover and integrate local sources of information, standards-based communication and rendering.

6. Conclusion

Ubiquitous mapping implies a flexible and fault-tolerant system of spatial data servers, mapping clients, and middleware that supports generation of online maps on multiple devices, for any requested location or theme. This paper argued that grid services and information mediation are promising approaches for ubiquitous mapping applications, and presented an outline of a scalable standards-compliant grid infrastructure for on demand interactive mapping on various clients including mobile devices. We introduced system-independent abstractions for composite mapping as declarative spatial integrated views specified in XQuery, and showed how two different implementations of map assembly service translate these mediator-generated views into a composite map.

While both map generation services described on the paper are operational, they are at different stages of maturity, as are other elements of the grid services-based mediation infrastructure. As grid interfaces gain acceptance, XQuery is extended to implement spatial join operations, more spatial data sources become available via standard web service calls, and systematic procedures for resolving semantic differences across sources get developed, ubiquitous cartography stands to benefit from universal on demand query access to multiple computer resources and map generation middleware capable of piecing together maps for different target devices.

Acknowledgements

Partial support under US National Science Foundation grants #0121269 "ITR/IM: Enabling the Creation and Use of GeoGrids for Next Generation Geospatial Information", #0205049 "ITR: GEON: The Geosciences Network: A Research Project to Develop Cyberinfrastructure for the Geosciences", as well as partial funding by ESRI and a University of California Discovery grant, are gratefully acknowledged

References

Baru, C., Gupta, A., Ludäscher, B., Marciano, R., Papakonstantinou, Y., Velikhov, P. and Chu, V. (1999). XML Based Information Mediation with MIX. Proceedings of the ACM SIGMOD 1999, pp. 597-599.

Boucelma, O., Esid, M., Lacroiz, Z. (2002) A WFS-based Mediation System for GIS Interoperability. Tenth ACM International Symposium on Advances in GIS, pp. 23-28.

Douglas, D., Poiker, T. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Canadian Cartographer 10:112–22.

Foster, I., Kesselman, C. and Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3).

Foster, I., Kesselman, C., Nick, J. and Tuecke, S. (2002) The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration (www.globus.org/research/papers/ogsa.pdf).

ESRI, 2004. ArcIMS and ArcXML references (www.esri.com).

GEON, the Geosciences Network (2003). www.geongrid.org

GGF (2003). Global Grid Forum (http://www.gridforum.org/)

Girow, A., (2002). TinyLine Project (http://www.tinyline.com/)

GLOBUS (2003). The GLOBUS Project (http://www.globus.org/)

Gupta, A., Marciano, R., Zaslavsky, I., Baru, C. (1999). "Integrating GIS and Imagery through XML-Based Information Mediation". In P. Agouris and A. Stefanidis (Eds.) *Integrated Spatial Databases: Digital Images and GIS*, Lecture Notes in Computer Science, Vol. 1737, pp. 211-234.

Lin, K., Ludäscher, B. (2003) A System for Semantic Integration of Geologic Maps via Ontologies,. In Semantic Web Technologies for Searching and Retrieving Scientific Data (SCISW), Sanibel Island, Florida, October 20th, 2003.

Manpuria, V., Zaslavsky, I., Baru, C. (2003) Web services for accuracy-based spatial query rewriting in a wrapper-mediator system. Presented at the Web and Wireless GIS (W2GIS) conference, Rome, December 2003, to appear in the post-conference proceedings.

OGC (2000). OpenGIS Web Map Server Interfaces Implementation Specification.

OGC (2001). OpenGIS, Geography Markup Language (GML) 2.0.

OGC (2002). OpenGIS Web Feature Service Implementation Specification.

Shimada, S., and Fukui, H. (1999) Geospatial mediator functions and container-based fast transfer interfaces in Si3CO Test-bed. LNCS 1580, pp. 265-276.

W3C (2001). Scalable Vector Graphics (SVG) 1.0 Specification, W3C Recommendation, 04 September 2001.

W3C (2003a). Web Services Description Language (WSDL) Version 1.2. W3C Working Draft 24 January 2003

W3C (2003b). Simple Object Access Protocol, W3C Proposed Recommendation, 07 May 2003.

W3C (2003c). Mobile SVG Profiles: SVG Tiny and SVG Basic, W3C Recommendation, 14 January 2003.

W3C (2004a). XQuery 1.0: An XML Query Language. W3C Working Draft 23 July 2004.

W3C (2004b) OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004

Wiederhold, G. (1992) Mediators in the Architecture of Future Information Systems. IEEE Computer, 25, 3, 38-49.

Zaslavsky, I. (2000). A New Technology for Interactive Online Mapping with Vector Markup and XML. Cartographic Perspectives, # 37 (Fall 2000), pp. 65-77.

Zaslavsky, I, Tran, J., Memon, A., Gupta, A., Ludäscher, B., Martone, ME (2002). AxioMap, and SVG based interfaces to a spatial query and markup system, Proceedings of the First International SVG Developers Conference, SVGOpen, Zurich, July 2002.

Zaslavsky, I., Memon, A., Petropoulos, M., and Baru, C. (2003) Online Querying of Heterogeneous Distributed Spatial Data on a Grid. Proceedings of the 3rd International Symposium on Digital Earth, pp. 813-823.

Zaslavsky, I., Baru, C., Bhatia, K., Memon, A., Velikhov, P., Veytser, V. (2003) Grid-enabled mediation services for geospatial information. Presented at the NSF Next Generation Geospatial Information (NG2I) workshop, Boston, October 2003 (forthcoming in LNCS.)

Zaslavsky, I, Memon, A. (2004). GEON: Assembling Maps on Demand From Heterogeneous Grid Sources. ESRI International User Conference (forthcoming).