Thesis Proposal

My Title suggestions based on the context of the document:

1. *Optimizing the responsiveness of the Distributed and Interoperable GIS*
2. *Designing interoperable and responsive Service Oriented GIS*
3. *Responsive and interoperable data rendering for Distributed GIS*
4. *Interoperable and Responsive Distributed GIS*
5. *Enhancing the responsiveness of Service Oriented GIS*

Ahmet Sayar
asayar@cs.indiana.edu

1. Introduction

Geographic Information Systems (GIS) [1, 2] is basically a collection of computer hardware and software for capturing, managing, analyzing, and displaying all forms of geographically referenced data.

General purpose of GIS is extracting information/knowledge from the raw geo-data collected from sensors, satellites or any other ways and stored in databases or file systems. The raw data goes through the filtering and rendering services and, geographically referenced information in human recognizable formats are produced. Perhaps the simplest example of GIS is map viewers which process layers of geospatial data to create map images. GIS are used in a wide variety of tasks such as urban planning, resource management, emergency response planning in case of disasters, crisis management and rapid response etc.

Over the past decades, GIS has evolved from traditional centralized systems to distributed systems [3]. Centralized systems provide an environment for stand-alone applications in which data sources, rendering and processing services are all tightly coupled and application specific. Therefore, they are not capable of allowing seamless interaction with the other data or processing/rendering services. These “deficiencies of centralized systems” and “improvements in the internet technologies” encouraged the academia, governments and businesses to start using distributed system approaches. Distributed systems are composed of autonomous hosts that are connected through a computer network. They enable sharing of data and computation resources, and collaboration on large scale applications.

However, there are some requirements for these distributed GIS applications to be efficient which can be categorized as functional and non-functional. Functional requirements are defined as accessing geospatial databases to execute spatially unified queries, and utilizing remote

geographic analysis, simulations, or visualization tools to process spatial data. On the other hand, there are several non-functional requirements such as interoperability and performance (or responsiveness). We will be focusing on the functional and these two non-functional requirements in our research.

Interoperability requirement is summarized as having a common service interfaces and data model enabling seamless inter-service communication. Distributed GIS has requirement of linking distributed data and rendering sources to create more complex and application specific information. In order for the services to be linked or to communicate, they need to be interoperable. In other words, they must know their interfaces and, how to invoke and process each other's responses. As a concrete example, our Web Map Server at CGL Lab might need to overlay NASA satellite maps with weather forecast information kept in a database of weather channel located in Seattle and, present it to the end-users.

Performance and responsiveness requirements can be briefly explained as getting the results in time. GIS which is used in emergency early-warning systems like homeland security and natural disasters (earthquake, flood etc) requires quick responses. However, because of the characteristics of geo-data (large sized and un-evenly distributed), time-consuming rendering processes and limited network bandwidth, the responsiveness of the system is one of the most challenging issues of the distributed systems.

In this thesis, we will first research architectural design requirements of an interoperable GIS framework (in accordance with Service Oriented Architecture (SOA)) composed of Web Service components of commonly accepted GIS Open standards. Secondly, we propose a novel grid-enabled [10, 11] aggregator map services addressing the performance and responsiveness issues. Our focus will be especially on the "view-level interoperability" and "responsiveness of the map rendering services". By the view-level we mean rendering raw data and overlaying over general maps or, composing layers from different rendering servers and creating more complex views.

2. Motivation

GIS systems require experts from different areas such as data maintenance and handling, data rendering and displaying, and data processing. Since it is hard to gather the necessary experts, data and hosts (such as high performance computers) in a geographically same place, it is inevitable to develop a interoperable distributed system architecture to enable coupling of these data and rendering/processing services remotely.

To create such an architecture, seamless communication and information exchange capabilities should be added to the interactions. Providing these capabilities improves Interoperability [8, 9] significantly**.** As a motivating example we can give layer-level data integration application in which layers are provided by heterogeneous Map Services having different service interfaces and deployed in different platforms. For example, if we want to build an Indiana State map showing all the counties, we have to use three different rendering services and three different client applications to communicate with. These are (1) ESRI's [5] ArcIMS and ArcMap Servers which

is used by Marion, Vanderburgh, Hancock, Kosciusko, Huntington and Tippecanoe counties, (2) Autodesk MapGuide [6] which is used by Hamilton, Hendricks, Monroe and Wayne counties, and (3) WTH Mapserver Web Mapping [7] which is used by Fulton, Cass, Daviess and City of Huntingburg counties.

Distributed GIS systems typically handle large volume of datasets. Therefore the transmission, processing and visualization/rendering techniques used need to be responsive to provide quick, interactive feedback. There are some characteristics of GIS services and data that make it difficult to design distributed GIS with satisfactory performance. One of them is that services provided by a GIS typically require heavy CPU usage due to the complex computation involved in the underlying computational geometry. Another one is that GIS services often transmit large resulting datasets such as structured data, images, or large files in tabular-matrix formats.

Our motivation is ensuring interoperable and responsive GIS system for the end users and Geo-science applications

3. Research Issues

In order to create interoperable distributed GIS and optimize its responsiveness, we will investigate following research issues.

The first one is creating interoperable GIS architecture. The interoperability challenges mostly result from data and service heterogeneity, and lack of wide-spread adopting of the universal standards. Service standardization includes first, defining the required standard online services for a complete system, and second, defining corresponding interfaces and message formats.

The second research issue is addressing the requirements for the performance and responsiveness of GIS involving large scale data transfer and processing. In most cases, the amount of collected data reaches to an amount in the order of gigabytes or even terabytes. Therefore, the GIS services must enable accessing and processing these large data sets in a reasonable time period. It even gets worse when the map animations and map movies (requiring many static map images to be created successively) need to be created.

Furthermore, due to the limited bandwidth and network speed, a GIS system faces the same performance problem as all the large scale distributed applications do. This challenge prevents making large scale geo-science applications feasible. Therefore, we will research the software level techniques such as parallel processing and caching on GIS domain.

In addition to aforementioned two main research issues we will also investigate developing a Web Services based SOA architecture with the universally accepted standards that provides access, querying, displaying and overlaying of data/information from map and feature services. Moreover, we will explore high-performance streaming data services through pub/sub based messaging system, composition of GIS services based on federation of service-capabilities, and coupling of geo- applications with archival data.

We identify the following research questions:

- How to build basic Web Service-based SOA architecture for GIS.
- How to incorporate widely accepted geospatial industry standards with Web Services.
- Are Web Services together with geospatial standards enabling acceptable performance?
- How to optimize the performance and responsiveness of the proposed GIS.
- How to make view-level data integration (aggregation).
- How to make responsive and unified querying of distributed data via their integrated views.
- How to apply parallel processing in order to enhance the GIS's responsiveness.
  -How to make range query partitioning and assembling the hierarchical data.

4. Methodology

Our proposed GIS architecture is Web Service-based Service SOA implemented in JAVA. In order to develop and evaluate our architecture, we chose Apache Axis 1.x [15] version to deploy Web Services. We exploited Apache Tomcat [14] as a servlet container. Apache Tomcat is developed in an open and participatory environment. It implements Java Server Pages (JSP) and the servlet specifications from Sun Microsystems.

In order to test the proposed system for performance and responsiveness, we develop interactive decision making tools enabling interactive display and querying of the data. For the creation of interactive browser-based system, we use AJAX (Asynchronous JAVA +XML), Dynamic HTML, JavaScript and Web Service client tools.

In order to achieve interoperable GIS system we use universally accepted Open Geosatial Consortium (OGC) [25] and ISO-TC211 standards specifications for online services and data model. As online standard services we develop Web Map Services (WMS) [12] and Web Feature Services (WFS) [13] (implemented by Aydin G.), and convert them into Grid-enabled Web Services by extensions.

5. Architecture in Brief

In order to investigate research issues mentioned in Chapter 3, we first propose an interoperable distributed GIS architecture (triangle in Figure 1) and then introduce innovative techniques to optimize the responsiveness of the system (WMS Aggregator in Figure 1).

Interoperability is obtained by using universally accepted OGC and ISOTC-211 standards. These entities define standards both for online services and for data models, which has been widely adopted in the GIS community. Even in the simplest GIS, there must be at least three major types of online services. These are basically data providing, data rendering and a display services. According to OGC standard definitions we develop Web Feature Service (WFS) as data service, Web Map Service (WMS) as rendering service (like filters), and interactive user portal as display service (see Figure 1). All these services are extended with Web Service standards to make

whole GIS system SOA. Developing Web Services-based GIS enable each individual services to be publish, located, invoked, and moreover, linked together to create more complex information.

As data model we use semi-structured GML [4] data defined by widely accepted OGC and ISOTC-211 standards. GML is basically an XML encoding for the modeling, transport and storage of geographic information including the spatial and non-spatial properties of geographic features. Feature is an entity related to earth with a geographic location such as road, river, states etc. GML has separate schema standards for both content part and presentation part. That enables display and query of the data. Display is related to presentation part of the data schema and query is related to attributes of the data schema.

There are two types of data flying around in the below GIS. One is for vector data encoded in GML common data model and provided to the system through WFS. GML encodes earth related features such as rivers, earthquake faults, seismic records, state boundaries etc in structured format. Another is binary data encoded in image types as map layers. These two data are handled differently.

Responsiveness and performance issues are mostly related to using structured common data model to solve the interoperability issues and limited bandwidth of the current internet. The key service proposed to handle these issues is WMS Aggregator (AWMS) (see Chapter 5.1) located at the top of GIS. It provides hierarchical data/information composition through its capability metadata. The general structure of the hierarchy (from top to bottom): project-> map -> layer -> {vector, raster and binary data}. It is actually a WMS with capability extensions such as: (1) map rendering in image layers, (2) information layering and layer-composition through capability federation (each service has capability in accordance with the schema defined by OGC), (3) request and response handlers covering query partitioning and assembling of the results returned to partitioned sub-queries. These are all handled according to the definitions and configurations in AWMS's capability metadata. User portal GUI is dynamically upgraded according to AWMS's capability metadata obtained through inter-service communication.

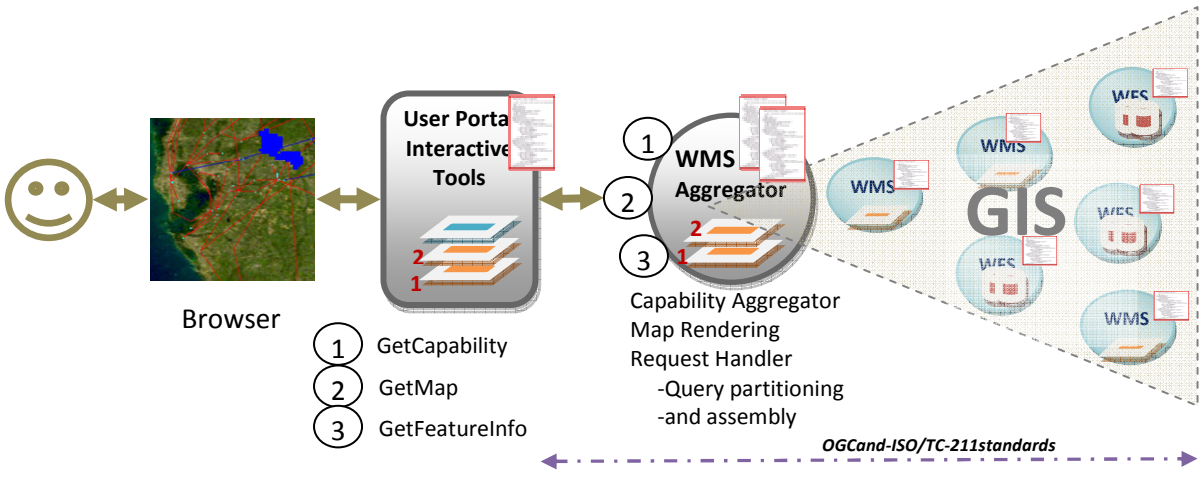

Figure 1: Proposed responsive and interoperable GIS architecture for end-users and Geo-science applications

Streaming data transfer for GML data traveling from WFS to WMS is implemented by using publish-subscribe based messaging middleware called Naradabrokering [18]. The clients make the requests with standard SOAP messages but for retrieving the results, a NaradaBrokering subscriber class is used. Through first request to Web Service (called *getFeature)*, WMS gets the topic (publish-subscribe for a specific data), IP and port to which WFS streams requested data. Second request is done by *NaradaBrokering* Subscriber.

Parsing and rendering of the structured GML data we use Pull Parsing technique. Pull parser only parses what is asked for by the application rather than passing all events up to the client application as SAX parsing does. The pull approach of this parsing model results in a very small memory footprint (no document state maintenance required – compared to DOM), and very fast processing (fewer unnecessary event callbacks - compared to SAX). You see the article where pull parsing is compared with other leading Java based XML parsing implementations [26].

The proposed GIS (triangle in Figure 1) is composed of chains of WFS and WMS services. All these services are described by their capability metadata (RDF-WSDL like semi structured schema defined by OGC) enabling inter-service communications.

WFS (implemented by Aydin G.) provides interfaces for data access on geographic data kept in geospatial database to retrieve and present the data to the clients in a standard data model (GML feature collections). WFS provide three major interfaces. These are “GetCapabilities”, “GetFeature” and “DescribeFeatureType”. WMS interact with WFS by submitting standard structured queries in compliance with OGC’s Filter Encoding and OGC Common Query Language.

WMS renders the raw binary or structured data captured from WFS and, creates comprehensible information in map images (MIME types JPEG etc). WMS defines a request/response protocol for web-based client/web-server interactions. It has three standard services, “GetCapabilities”, “GetMap" and “GetFeatureInfo”. Communications between WMS and WFS is achieved by exchanging the capability documents through their GetCapabilities service interface. Capability document include all the required information to make successive request to use other service interfaces such as GetFeature, GetMap and GetFeaturInfo.

GetCapabilities request allows the server to advertise its capabilities such as available layers, supported output projections, supported output formats and general service information. After getting this request, WMS returns an XML document with the metadata about the WMS Server. This capability file is kept in the local file system.

The getMap service interface allows the retrieval of the map. getMap request is consist of attribute value pairs. Attributes are predefined in a standard form but values are assigned in accordance with the capability document of the server invoked. The most of the WMS clients are browser based and GUI enabled. Values for the attributes are assigned with the help of user interface and interactive map tools such as zoom-in, zoom-out and panning.

GetFeatureInfo is for making unified querying over the feature data used for creating the map layers. It is used for querying the attributes (content) of the data. The common data model used

to encode feature data in the system (GML) enables this. GML has two parts, one is content and the other is presentation. The presentation part is for rendering and displaying geometry elements such as points, line-strings, polygons etc. Content part is related to the attributes of the data such as the magnitude values of seismic records and/or length of a fault segment.

5.1. WMS Aggregator (AWMS)

AWMS is actually a WMS with some extensions providing enhanced map rendering services by using innovative pre-fetching, parallel processing and caching techniques. AWMS aggregates, composes and orchestrates WMS and WFS services and, express the layer level compositions in its capabilities file by federating other services' capabilities metadata, and present its capability to the user portal (or client) with "GetCapabilities" Web Service interface.

In summary, pre-fetching is purely for overcoming the natural bandwidth problem, caching helps system prevent redoing the jobs of querying and rendering before, and parallel processing helps workload sharing and parallel job run. Depending on the data characteristics, AWMS uses only one or the combination of these techniques.

Pre-fetching is for rendering of the data - not changing often. Pre-fetching is getting the data before it is actually needed and keeping it in local server for the successive requests done for the same data. In our framework we use archived data and they don't change frequently. Therefore, it is not reasonable to transfer and render the same data repeatedly for every request with minor changes in their request criteria. We use pre-fetching as a solution to this type of problem. Pre-fetching will be done for GML data transfers between WMS and WFS based on the predefined periodicity. Periodicity is defined depending on the data's characteristics, ex. once a day.

Sometimes pre-fetching is not efficient because of the characteristics of the data such as changing frequently and limited storage capacity of the initiating server. In these cases, we use parallel processing approach applied together with caching.

Caching is composed of two types of processes. These are "cached-data extraction" and "rectangulation" of remaining area in main query (see *Figure 2*). We utilize from cached data by extracting the overlapped region and prepare request for the remaining parts through *rectangulation* processes. AWMS assembly the extracted cached data with the returned map images returned as responses to sub-queries corresponding to rectangles obtained previously through the rectangulation process.

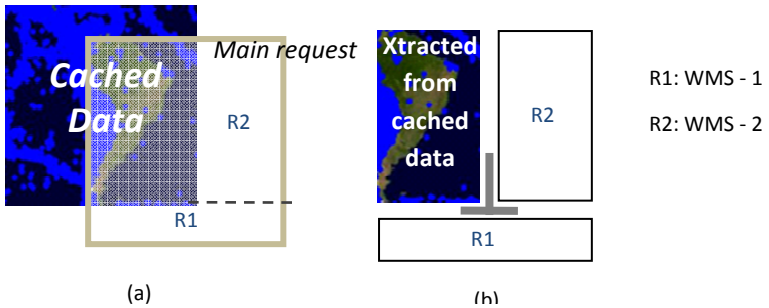

Figure 2: (a) cached data extraction (b) parallel processing through caching, query partitioning and rectangulation

AWMS apply the **parallel processing** after cached data extraction and rectangularion. Since all the data in the system is geo-referenced, and queried in ranges defined by bounding boxes (defining coordinates of rectangles in the form of (minx, miny, maxx, maxy)), we do range query partitioning to implement parallel processing. Partitioning the queries for parallel processing can be considered in two ways. One is layer-based (horizontal) another is partitioning the bbox criteria of the queries. These issues are taken cared of at the AWMS and layer compositions and partitioning are configured in its capability metadata.

As it is mentioned earlier, parallel processing is done for handling frequently changing data. We will research what techniques give better results for what type of data and in what conditions by doing benchmarking.

6. Benchmarking

After we are done with the architecture, we integrate the system with the real Geo-science applications such as Pattern Informatics (PI) [16] and Virtual California (VC) [17] through other CGL Lab projects such as WS-Context [27] and HPSearch [28].

In order to address research issues we will perform performance-responsiveness tests. We are planning to do below performance tests which are illustrated in Figure 3:

Each test will be done on a single machine, local area network and wide area network separately.

For different data size

- Timing for rendering and data display ***(T)***
  - *Parsing and rendering a map from GML data locally in WMS*
  - Caching and parallel processing (for the frequently changing data)
  - Pre-fetching (for the rarely changing data)
- Timing for data (GML) transfer issues (from WFS to WMS –see Figure 3) $(t_{1.1}\text{-}t'_{1.3})$
  - Streaming data transfer (through pub/sub based messaging middleware [18])
  - Non-streaming data transfer
- Stress tests:
  - Streaming map movies requiring high performance data transfer and rendering.
  - Querying, rendering and displaying data in GBs (take hours – need in seconds)

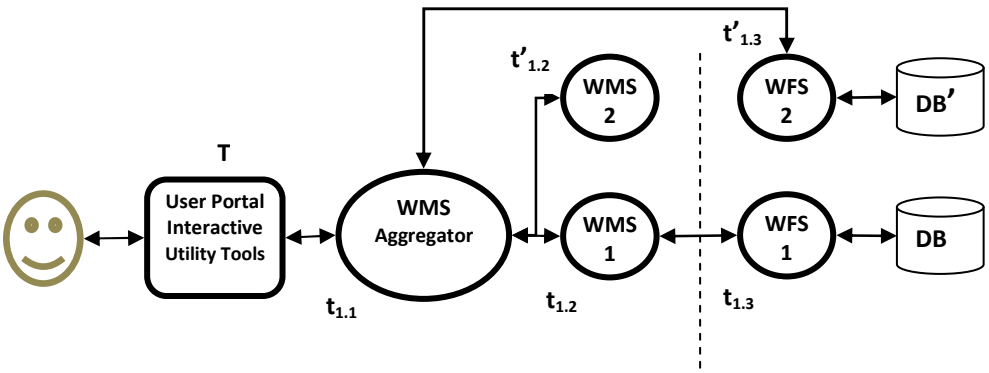

Figure 3: Performance test illustrations.

Measurement of success: We will compare the performances of the system with the other well-known deegree project [19] and possibly UMN MapServer [20] map-rendering tools. These are the two best well-known scientific map rendering services developed with the universally accepted standards.

7. Related Work

Linked Environments for Atmospheric Discovery (LEAD) [21] is a large scale project funded by NSF Large Information Technology Research grant for addressing fundamental IT and meteorology research challenges to create an integrated framework for analyzing and predicting the atmosphere [23]. On the other hand we aim same things for the earth-related Geo-science. LEAD and our architecture both use SOA for the utilization of distributed sources.

At the application level, LEAD supports for adaptive analysis and prediction of large scale meteorological events. They call it forecast mode using available observations or model generated data and manages necessary resources. They mostly focus on automated data management, scalable data archiving system and easy search and access interfaces via GUI and underlying ontology. LEAD has MyLEAD concept for enabling users to process their own data. Users can interactively explore the weather as it evolves, create custom scenarios or acquire and process their own data. Our approach is based on accessing querying and displaying the data not management.

As we do they also provide a web-portal as the entry point for students, users or advanced researchers to the meteorological data, services, models, and workflow, analysis and visualization tools related to the project.

Like LEAD, GEON [22] is also SOA based architecture adopted with a portal developed for end-users and, NSF funded large scale project involving the development of a distributed, services-based system that enables geoscientists to publish, integrate, analyze, and visualize their data.

In summary, compared to the related projects our contributions are: (1) Introducing the innovative techniques for high performance and responsive map-data rendering, (2) interactive and seamless data access and querying architectures and, (3) developing interoperable GIS via SOA and universally accepted standards (OGC, ISO-TC211 and W3C).

8. Work Plan

(1) Developing streaming and non-streaming versions of Web Map Services in accordance with universally accepted standards. (2) Extending WMS as Web Services –grid-enabled (3) Developing WMS Aggregator from basic WMS. (3) Extending OGC standards to enable pre-fetching, parallel processing and caching. (4) Integrating the system with real Geo-science applications such as PI and VC applications and other CGL lab projects such as WS-Contect and HPSearch. It covers developing the interactive web-based map tools to interact with the coupling of real Geo-Science applications and map animation tools such as streaming map movies. (5) The performance and responsiveness tests and evaluation of the proposed system.

References

[1] GIS Research at Community Grids Lab, Project Web Site: http://www.crisisgrid.org.

[2] Ahmet Sayar, Marlon Pierce, Geoffrey Fox OGC Compatible Geographical Information Services Technical Report (Mar 2005), Indiana Computer Science Report TR610

[3] Peng, Z.R. and M. Tsou, Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks. 2003: Wiley

[4] Cox, S., Daisey, P., Lake, R., Portele, C., and Whiteside, A. (eds) (2003), OpenGIS Geography Markup Language (GML) Implementation Specification. OpenGIS project document reference number OGC 02-023r4, Version 3.0.

[5] ESRI, ArcIMS, 9 Architecture and Functionality, J-8694. ESRI White Paper, http://downloads.esri.com/support/whitepapers/ims_/arcims9-architecture.pdf. 2004.

[6] Autodesk. MapGuide http://usa.autodesk.com. [cited.

[7] MapServer, W. http://www.wthengineering.com/GIS/web_gis.htm. [cited.

[8] de La Beaujardiere, J., Web Map Service, OGC project document reference number OGC 04-024. 2004.

[9] Vretanos, P. (2002) Web Feature Service Implementation Specification, OpenGIS project document: OGC 02-058, version 1.0.0. Volume,

[10] Fox, G. and M. Pierce. Web Service Grids for iSERVO. in International Workshop http://www.eps.s.u-tokyo.ac.jp/jp/COE21/events/20041014.pdf on Geodynamics: Observation, Modeling and Computer Simulation University of Tokyo Japan October 14 2004. 2004

[11] Fran Berman, Geoffrey C, Fox, Anthony J. G. Hey., Grid Computing: Making the Global Infrastructure a Reality. John Wiley, 2003.

[12] de La Beaujardiere, J., Web Map Service, OGC project document reference number OGC 04-024. 2004.

[13] Vretanos, P. (2002) Web Feature Service Implementation Specification, OpenGIS project document: OGC 02-058, version 1.0.0. Volume,

[14] Apache Tomcat, http://tomcat.apache.org/.

[15] Apache Axis, http://ws.apache.org/axis/.

[16] Tiampo, K. F., Rundle, J. B., McGinnis, S. A., & Klein, W. Pattern dynamics and forecast methods in seismically active regions. Pure Ap. Geophys. 159, 2429-2467 (2002).

[17] Rundle, P.B, J.B. Rundle, K.F. Tiampo, A. Donnellan and D.L. Turcotte, Virtual California: Fault Model, Frictional Parameters, Applications, PAGEOPH, submitted

[18] Pallickara, S. and G. Fox. NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. in Lecture Notes in Computer Science. 2003: Springer-Verlag.

[19] Deegree projects home page http://deegree.sourceforge.net/

[20] UMN MapServer project home page http://mapserver.gis.umn.edu/

[21] Beth Plale, Dennis Gannon, Dan Reed, Sara Graves, Kelvin Droegemeier, Bob Wilhelmson, Mohan Ramamurthy, "Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD", *To appear ICCS workshop on Dynamic Data Driven Applications*, Atlanta, Georgia, May 2005.

[22] GEON (Geosciences Network): A Research Project to Create Cyberinfrastructure for the Geosciences. http://www.geongrid.org

[23] Kelvin K. Droegemeier, et al. Linked environments for atmospheric discovery (LEAD): A cyberinfrastructure for mesoscale meteorology research and education. in 20th Conf. on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology, . 2004. Seattle, WA.

[24] Sosnoski, D. "XML and Java Technologies", performance comparisons of the Java based XML parsers. Available at http://www-128.ibm.com/developerworks/xml/library/x-injava/index.html

[25] OGC (Open Geospatial Consortium) official web site http://www.opengeospatial.org/

[26] Sosnoski, D. "XML and Java Technologies", performance comparisons of the Java based XML parsers. Available at http://www-128.ibm.com/developerworks/xml/library/x-injava/index.html

[27] Bunting, B., Chapman, M., Hurlery, O., Little M., Mischinkinky, J., Newcomer, E., Webber J, and Swenson, K., Web Services Context (WS-Context) Specification, Version 1.0, July 2003.

[28] Harshawardhan Gadgil, Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Robert Granat, Proceedings of the IEEE/ACM Cluster Computing and Grid 2005 Conference, CCGrid 2005, Cardiff, UK