

DEVELOPING GIS VISUALIZATION WEB SERVICES FOR GEOPHYSICAL APPLICATIONS

A. Sayar^{a,b,*}, G. C. Fox^{a,b,c}, M. Pierce^a

^a Community Grids Laboratory, Indiana University
501 N. Morton Suite 224, Bloomington, IN 47404
{asayar, gcf, mpierce}@cs.indiana.edu

^b Computer Science Department, School of Informatics, Indiana University

^c Physics Department, College of Arts and Sciences, Indiana University

Commission II, WG II/2

KEY WORDS: GIS, Geophysics, Visualization, Internet/Web, Interoperability, Networks, Distributed.

ABSTRACT:

The Open Geospatial Consortium (OGC) defines a number of standards (both for data models and for online services) that have been widely adopted in the Geographical Information System (GIS) community. In this paper we will describe our group's efforts to implement GIS services according to OGC standard specifications in accordance with the Web Services approach. This paper focuses on the Web Map Service (WMS), which we are coupling to problems in computational geophysics. Through the use of Web Services, we are able to integrate GIS services with other families of services, including information, data management, and remote application execution and management. We also describe WMS client building efforts that are suitable for integration with computational Web portals.

To be able to interact with non-Web Service versions of WMS, we have built bridging service for our extended WMS. Since Web Service oriented WMS has a different request/response paradigm from non-Web Service versions, we have extended cascading WMS by adding request handler functionality. This kind of WMS behaves like both a cascading WMS and a proxy to handle different types of requests to overcome interoperability problems between different WMS systems.

1. INTRODUCTION

Geographical Information Systems (GIS) introduce methods and environments to visualize, manipulate, and analyze geospatial data. The nature of the geographical applications requires seamless integration and sharing of spatial data from a variety of providers. Interoperability of services across organizations and providers is a main goal for GIS and also Grid computing [15, 23].

To solve the interoperability problems, the Open Geospatial Consortium (OGC) has introduced standards by publishing specifications for the GIS services. OGC is a non-profit, international standards organization that is leading the development of standards for geographic data related operations and services. OGC has variety of contributors from different areas such as private industry and academia to create open and extensible software application programming interfaces for GIS [1].

GIS services, such as defined by the OGC, are part of a larger effort to build distributed systems around the principles of Service Oriented Architectures (SOA). Such systems unify distributed services through a message-oriented architecture, allowing loose coupling, scalability, fault tolerance, and cross-organizational service collections [21]. Web Service standards [3] are a common implementation of SOA ideals, and Grid computing has converging requirements [15, 23]. By implementing Web Service versions of GIS services, we can integrate them directly with scientific application grids [11, 30, 32].

This document gives the details about the design and architecture of our Web Service refactoring of OGC specifications for the Web Map Service. This is part of a larger effort by our group to investigate translations of GIS services into Web Service standards [2, 22]. Some earlier work in this area is reported in WMS [13]. In these documents they define standard WSDL description of the service interfaces.

* Corresponding author.

In this document we first give some background information and explain some related works. In this section we compare our basic GIS services with OGC Web Services Specifications published lately. Section 3 explains Web Services technologies and advantages from the point of GIS view. In Section 4, we describe general architecture for developing Web Service compatible mapping services. Under this title as subtopics we describe contributions of the Web Services to the GIS services, technical challenges encountered during implementations, integrating Web Services into OGC compatible GIS visualization, creating valid requests to WMS services in case of using Web Services, bridging capability of cascaded WMS, other services involved in proposed visualization system and implementation details of WMS. Section 5 describes our generic and modular WMS client implementation for the geophysics applications and gives a sample case scenario. In Section 6, future work is described. Section 7 is the conclusion.

2. BACKGROUND

After Web Services gain momentum and wide acceptance, some entities working on GIS started to involve in Web Services area and tried to integrate and/or convert their GIS servers and applications into web services. Some of these entities are commercial GIS leading companies such as ESRI, Cubewerx, Demis and Intergraph. ESRI produced ArcWeb service package for the GIS Web Services by using UDDI for the catalog and registry services. Cubewerx, Demis and Intergraph provide WMS transparent access to their Web Service mapping applications.

OGC as a GIS standards body published its Web Services Common Implementation Specifications lately. We have some differences with the latest OGC Web Service Specifications. In our implementation of GIS Web Services, return types defined as Strings for the WMS `getCapabilities` and `getFeatureInfo`. Returned strings are structured data in XML. Strings are actually xml, plain text, HTML or GML depending on the requested format. In case of `getMap` request WMS returns image MIME type such as `image/jpeg` as `DataHandler` object attached to SOAP message. OGC has different return types defined for the different types of requests. Our implementation for the return types will be improved and changed in accordance with OGC Web Services Specifications. When we first started to implement GIS Web Services OGC did not have this new specification. We will be adapting our request response object to their schemas.

Regarding to catalog registry services, instead of using OGC WRS (Web Registry Services) [34] we utilize an alternative Registry Information Model; the Universal Description, Discovery, and Integration (UDDI) [29]. UDDI is domain-independent standardized method for publishing/discovering information about Web Services. As it is WS-Interoperability (WS-I) compatible, UDDI has the advantage being interoperable with most existing Grid/Web Service standards. WRS is an OGC standard to discover/publish service information of geospatial services. It presents a domain-specific registry capability for geospatial information. UDDI is domain-independent standardized method for publishing/discovering information about Web Services.

Our approach to catalog registry services in GIS applications takes into account the descriptive metadata, i.e. quality of service

attributes, into discovery process. The geospatial data being provided by a geospatial service may be fitted with client's request, however, this does not necessarily guarantee whether the service is sufficient for the desired quality of service requirements. By matching Quality of Service attributes of service discovery request and service descriptions, client is able to distinguish geospatial services that match to their requirements.

Regarding to our Web Service based WFS, the format of the request and response objects is String in the form of XML. Each request and response has its own schema file. They are created according to these schema files and given parameters. After creating objects as XML, in the Web Service GIS environment, XML objects are put into extensible SOAP envelope. Requests and responses are carried in the SOAP message over the HTTP. For the architecture details please see the Section 4

3. WEB SERVICES FOR GIS

Web Services give us a means of interoperability between different software applications running on a variety of platforms. Web Services support interoperable machine-to-machine interaction over a network. Every Web Service has an interface described in a machine-readable format. Web Service interfaces are described in a standardized way by using Web Service Description Language (WSDL) [17]. WSDL files define input and output properties of any service and services' protocol bindings. WSDL files are written as XML documents. WSDL is used for describing and locating Web Services. Web Services are defined by the four major elements of WSDL, "portType", "message", "types" and "binding". Element `portType` defines the operations provided by the Web Services and the messages involved for these operations. Element `message` defines the data elements of the operations. Element `types` are data types used by the Web Service. Element `binding` defines the communication protocols. Other systems interact with the Web Service in a manner as described in WSDL using Simple Object Access Protocol (SOAP) messages.

SOAP [16] is an XML based message protocol for exchanging the information in distributed environment. It provides standard packaging structure for transporting XML documents over a variety of network transport protocols. It is made up of three different parts. These are the envelope, the encoding rules and the Remote Procedure Call (RPC) convention. SOAP can be used in combination with some other protocols such as HTTP. OGC compatible Web Services will be using SOAP over HTTP.

Advantages of the Web Services in GIS area can be grouped into three categories:

Distribution: It will be easier to distribute geospatial data and applications across platforms, operating systems, computer languages, etc. They are platform and language neutral.

Integration: It will be easier for application developers to integrate geospatial functionality and data into their custom applications. It is easy to create client stubs from WSDL files and invoke the services.

Infrastructure: We can take advantage of the huge amount of infrastructure that is being built to enable the Web Services

architecture – including development tools, application servers, messaging protocols, security infrastructure, workflow definitions, etc [13]. Some of these features are being developed by using Web Service infrastructure in Naradabrokering [24], message based middleware system, developed in CGL (Community Grids Lab.) at Indiana University. NaradaBrokering aims to provide a unified messaging environment that integrates grid services, web services, peer-to-peer interactions and traditional middleware operations. In the near future we will be utilizing these features in GIS visualization systems.

GIS services can be grouped into three different categories; these are data services, processing services and registry, or catalog services [33]. Data services are tightly coupled with specific data sets and offer access to customized portions of the data. Processing services provide operations for processing or transforming data in a manner determined by user-specified parameters. Registry or catalog services allow users and applications to classify, maintain, register, describe, search and access information about Web Services. In our development of GIS web services for the geophysical applications, we use WFS as data services, IS as catalog-registry services and WMS as processing services. For the architecture details see the Section 4.

4. ARCHITECTURE

This section gives the details of the integrations of Web Services technologies into OGC compatible GIS visualization Systems.

We first implemented pure OGC compatible WMS and WFS servers. These servers were communicating over HTTP by making HTTPGET/POST requests. Later, we developed a generic conversion algorithm steps for converting HTTP based visualization systems into service based counterparts. These steps are listed below;

1. Define a WSDL for the OGC Web Services (OWS) as a set of interface definitions for its functionalities.
2. Create appropriate XML Schema for all the requests and responses that OWS provides. These schemas are created according to the attributes and properties of OGC OWS specifications.
3. Create client stubs from the WSDL file of the target OWS.
4. After creating stand-alone Web Services compatible OGC GIS server, you are ready to bridge this kind of server to other generic OGC servers. (See Section 4.4 for the WMS case)

As a case study we worked on WMS to apply these conversion algorithms. For the step-1 you can create service interface (WSDL) by using some web service tools. Before doing that you need to implement the functionalities in any language as you did in HTTP based GIS services. For getting details for the other steps please see the Section 4.1 and 4.4. Section 4.2 explains the other components in the visualization systems, Section 4.3 gives the architecture and implementation details of the main visualization service, WMS.

4.1 Creating valid Request to WMS Services in Case of Using Web Services

In developing Web Service versions of the WMS, we have converted existing HTTP GET/POST conventions [4] into WSDL interfaces. We have encountered some minor technical problems in this conversion.

Internal implementations of the WMS services are compatible with the current WMS specifications but service interfaces and the way to invoke services are different. Services are invoked through the SOAP over HTTP. Requests are created as XML documents and wrapped into body part of the SOAP request message. These requests are shown in Figure 1-3.

Invoking WMS operations should be according to specifications. OGC compatible requests to WMS are well defined in the WMS specifications [4]. Requests must have some parameters whose names, numbers, and values should obey the rules defined in the specifications to be OGC compatible. In this section we define these requests in the form of XML schema files.

These schema files are created to be used during the invocation of the operations implemented as Web Services at the WMS side [13]. Requests are created at the WMS Client side. Clients create these requests after getting required parameter from the user. When request is ready, client sends this request to WMS as a SOAP message. WMS has deployed Web Services for each service, getMap, getCapabilities and getFeatureInfo. Clients use client stubs created before to invoke these specific Web Services. All these services in WMS take one String parameter. This String parameter is request itself. These requests are actually XML documents in String format.

Below schema files displayed in Figure 1-3 include all the elements and attributes of corresponding OGC HTTPGET/POST requests defined in OGC WMS specifications [4].

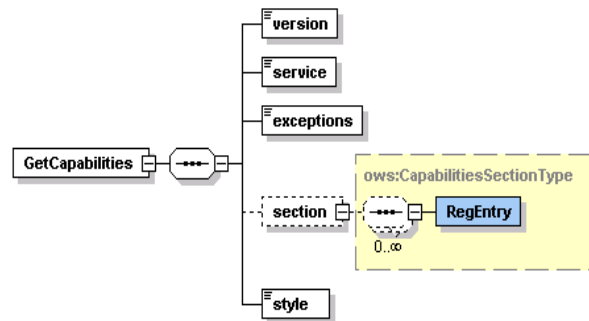


Figure 1 : GetCapabilities Request Schema.

GetMap request is created for our WMS implementation. We have not implemented styling capability yet. Styling capability will be added soon, for the current status and the future works please see the Section 6. WMS supporting styling are called SLD-enabled WMS. The Open GIS Consortium (OGC) Styled Layer Descriptor (SLD) specification [6] defines a mechanism for user-

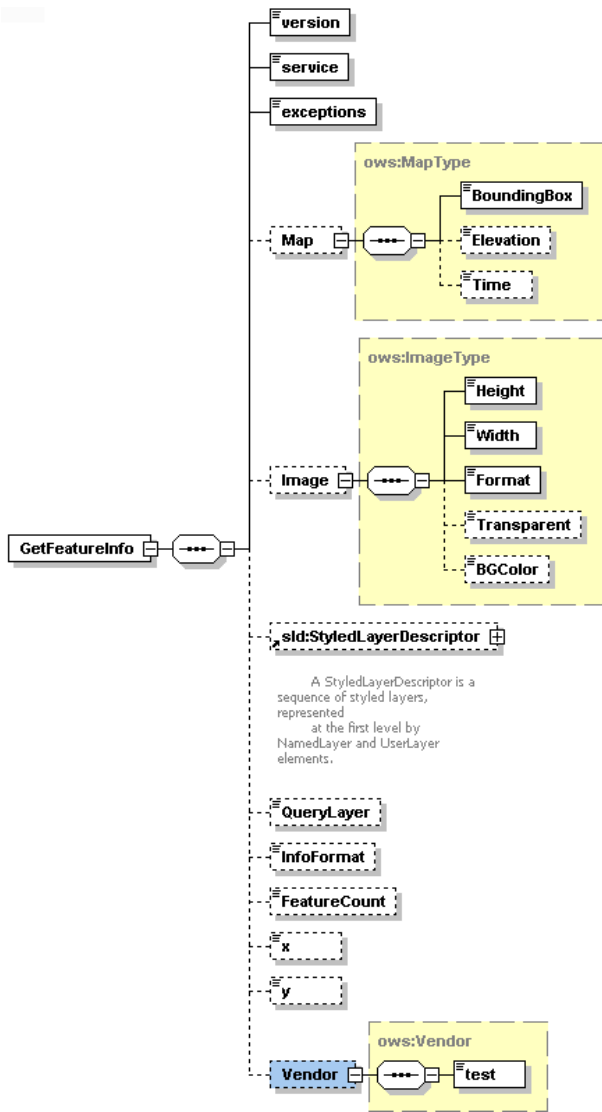


Figure 2 : GetFeatureInfo Request Schema.

defined symbolization of feature. An SLD-enabled WMS retrieves feature data from a Web Feature Service [7] and applies explicit styling information provided by the user in order to render a map.

In our project since we have just implemented Basic WMS, we have not used elements related to styling in the WMS getMap requests. For defining styling in the getMap request we use StyledLayerDescriptor element. StyledLayerDescriptor has other sub elements and attributes.

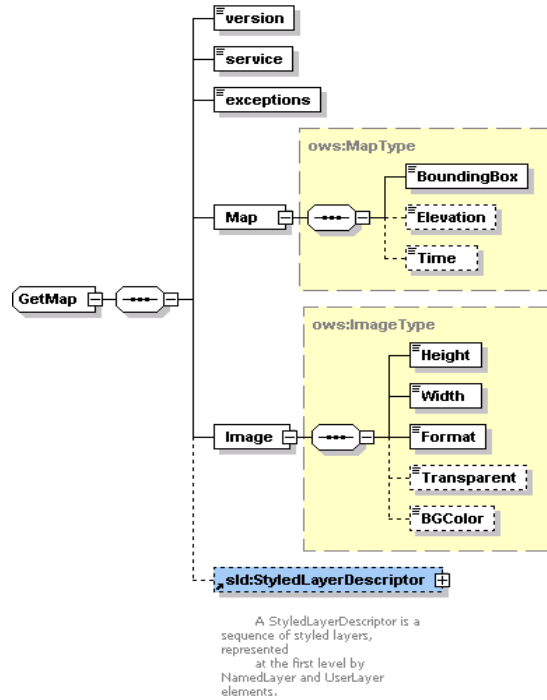


Figure 3 : GetMap Request Schema.

4.2 Other GIS Components Involved in Proposed Visualization System

Our Web Service-compatible WMS depends upon Web Feature Service [27] and (IS) Information Services [28] to accomplish its required tasks. They are ongoing projects in CGL (Community Grids Lab.). This section briefly describes the WMS interactions with these other services.

A general picture of interactions between these three services is displayed in Figure 4. Initial invocations are displayed as black arrows. All the services are implemented as Web Services.

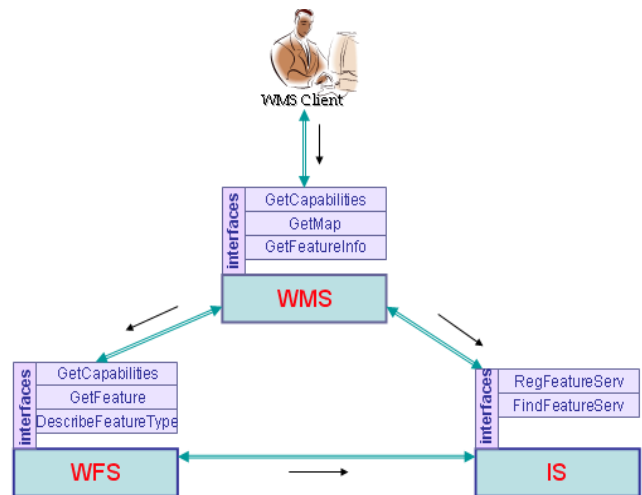


Figure 4 : Basic GIS Components Involved in Visualization System.

4.2.1 Web Feature Service (WFS): WFS instances store geospatial data and serve them upon request from clients. WFS clients include Web Map Servers and other WFS instances (in case of cascading WFS). WFS provide feature vector data. Vector data are encoded in GML (Geographic Markup Language) [9], an XML encoding for the transport and storage of geographic information, including both the geometry and properties of geographic features.

According to OpenGIS WFS specification, basic Web Feature Services are getCapabilities, describeFeatureType and getFeature. If WFS is transactional than this WFS provides two more services. These are “transaction” and “lockFeature” services. Our implementation of WFS is basic WFS, so it does not have transaction and lockFeature capabilities.

Since we have implemented basic WFS, WMS uses basic WFS services: getCapabilities, describeFeatureType, and getFeature. WMS sends a getCapabilities requests to WFS to learn which feature types WFS can service and what operations are supported on each feature type. The getCapabilities request can also be mediated by the aggregating Information Services (IS). WMS makes its request to IS to get a specific WFS address that provides needed feature. Please see Section 4.2.2 for the details about the interconnection between WMS and IS.

When any WMS client sends a getFeatureInfo request to WMS, WMS creates a getFeature request and sends it to WFS. The URL address of the WFS is found by using IS. After choosing appropriate WFS, the WMS makes a getFeature requests to get feature data. A sample request is shown in Figure 5 . The GML file encoded in XML is returned in a SOAP envelope as a response to this request.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<wfs:GetFeature outputFormat="GML2"
xmlns:gml="http://www.opengis.net/gml"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:ogc="http://www.opengis.net/ogc">
  <wfs:Query typeName="boundary_lines">
    <wfs:PropertyName>FNODE</wfs:PropertyName>
    <wfs:PropertyName>TNODE</wfs:PropertyName>
    <wfs:PropertyName>WORLD</wfs:PropertyName>
    <wfs:PropertyName>coordinates</wfs:PropertyName>
    <wfs:PropertyName>minx</wfs:PropertyName>
    <wfs:PropertyName>miny</wfs:PropertyName>
    <wfs:PropertyName>maxx</wfs:PropertyName>
    <wfs:PropertyName>maxy</wfs:PropertyName>
    <ogc:Filter>
      <ogc:BBOX>
        <ogc:PropertyName>coordinates</ogc:PropertyName>
        <gml:Box>
          <gml:coordinates>-155,28 -120,50</gml:coordinates>
        </gml:Box>
      </ogc:BBOX>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>

```

Figure 5 : Sample GetFeature Request from WMS to WFS.

4.2.2 IS (Information-Discovery Services): An OGC Catalog [14] is a collection of descriptive information (metadata) regarding the data stored in a geographic database. OGC catalog service is specific to OGC domain. Each GIS Service provides access to geographic data. An important factor that characterizes GIS Services is the metadata about the data. Thus, metadata act as

properties that can be queried and requested through catalog services. A catalog service provides discovery of GIS services through the metadata of the data that these services provide. The OGC Catalog Service provides useful GIS metadata and registry capabilities, but we are interested in making several extensions. For instance, the registry should also allow discovery of services based on non-functional requirements of services such as Quality of Service attributes. Also, OGC Catalog Service should be consistent with other existing and more general registry models such as UDDI or ebXML.

To overcome these limitations, we utilize a Registry model which is being developed in CGL as a general registry model for Web Services, Fault Tolerant High Performance Information Services (FTHPIS) [28]. IS is a general service registry and discovery model based on UDDI specifications [29]. UDDI is WS-I approved specifications, in other words, it is inter operable with other Web Service based standards. An IS provides both publishing and discovery services for Web Services and (WS-Context) [19] contextual information of GIS Services. Since IS stores both functional metadata (metadata about GIS data) and non-functional metadata (metadata about Quality of Services of data, such as high throughput), it provides more complex query abilities when discovering GIS services.

A map server interacts with Information Services to dynamically discover available Web Feature Services. We can summarize the interaction between an Information Service, Web Feature Service and Web Map Server as following.

All GIS Web Feature Services are expected register themselves into an existing IS in order to be "discoverable". Once the registry is completed, the IS starts interacting with WFS to retrieve more information about their capabilities. So, IS stores information about the functionalities of each WFS.

A Web Map Server queries an Information Service to find available WFS. Apart from discovery of the services, WMS can create capabilities file of a WFS on the fly, as the IS provide extensive information about the capabilities of WFS. An IS provides consistent and uniform API for publishing and discovering OpenGIS Web Services, and it is defined by a WSDL. Once the WFS are dynamically discovered through IS, WMS can then invoke corresponding WFS to retrieve the features that it needs.

4.3 Visualization Service – WMS

WMS is the key service to the GIS visualization system. WMS produce maps from the geographic data. A map is not the data itself. Maps create information from raw geographic data, vector or coverage data. Maps are generally rendered in pictorial formats such as jpeg (Joint Photographic Expert Group), GIF (Graphics Interchange Format), PNG (Potable Network Graphics). WMS also produce maps from vector-based graphical elements in Scalable Vector Graphics (SVG) [18].

WMS provide three main services; these are getCapabilities (Section 4.3.1), getMap (Section 4.3.2) and GetFeatureInfo (Section 4.3.3). GetCapabilities and getMap are required services to produce a map but GetFeatureInfo is an optional service. These services and our implementations are explained in the following subsections.

4.3.1 GetCapabilities from WMS: Before a WMS Client requests a map from WMS, it should know what layers WMS provides in which bounding boxes. GetCapabilities request enables WMS Clients to obtain this type of information about the contacted WMS. 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 capabilities file is kept in the local file system and sent to clients upon getCapabilities request.

After getting the request WMS parses it to derive parameters. If WMS verifies that request, then it sends the capabilities file to the WMS Client as a SOAP attachment. If WMS encounters any problem during handling of the request than it sends exception message in SOAP back to the WMS Client. Basic getCapabilities request are pictured out at Figure 6.

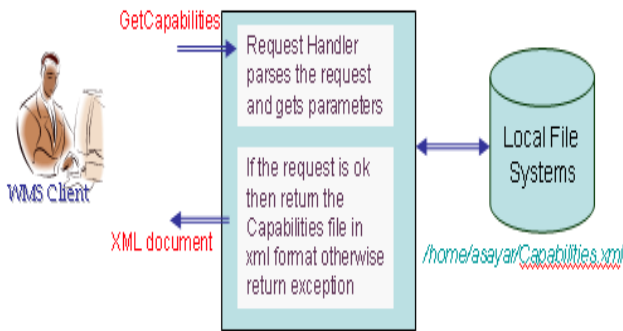


Figure 6 : getCapabilities work flow.

4.3.2 GetMap from WMS: Another service interface that WMS provides is GetMap request. The getMap service interface allows the retrieval of the map. Chained processes to produce maps are illustrated in Figure 7. This request is done by the client after finishing getCapabilities request and defining the available layers. After getting the getMap request, the WMS goes over the flow depicted in Figure 7 and if everything succeeds, then returns the result as an image in a format defined in the getMap request. All the supported image formats are defined in WMS Capabilities document. Requests for the image formats should be made in accordance with the WMS's Capabilities file. The image is returned back to the WMS Client as an attachment to SOAP message. If the WMS encounters any problem during handling of the request, it sends an exception message in SOAP back to the WMS Client.

WMS first parses the parameters and get their values from the getMap. Depending on these parameters, WMS might need to make some requests to some other WMS services. WMS first determines what layers are requested, in which bounding box, in which form, and so forth. After determining all the request parameters, it makes find_service and getAccess_point requests to IS to determine the WFS providing requested feature data. These requests are done as SOAP messages to IS service interfaces implemented as Web Services. GetAccess_point returns the Web Service access point address of the WFS that provides the

requested feature. WMS makes getFeature request to the returned WFS and gets the requested feature data in GML format. If the parameter defining returned image format in getMap request is Scalable Vector Graphics (SVG), then WMS creates SVG from returned feature data by using its geometry elements. If the requested image is not in SVG format, we first create the SVG image and then convert it into the desired image formats (such as PNG, GIF, or JPEG). Apache Batik provides libraries for this conversion. Batik is a Java(tm) technology based toolkit for applications or applets that use images in the SVG format for various purposes, such as viewing, generation or manipulation. By using these schema files we derive geometry elements from the GML file to visualize the feature data. These geometry elements in GML [9] are basically Point, Polygon, LineString, LinearRing, MultiPoint, MultiPolygon, MultiGeometry, etc.

To create the images from the features returned from the WFS, we have used Java Graphics2D and Java AWT libraries. For each layer we create a different graphics object. If you assign each layer to different graphics object than Java libraries allow you to overlay these graphic objects.

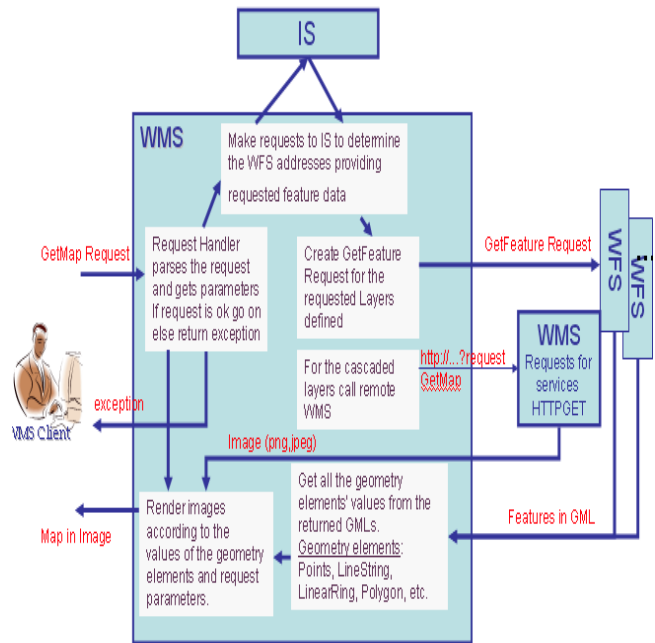


Figure 7 : getMap work flow.

4.3.3 GetFeatureInfo from WMS: This is an optional WMS service. It is not necessary to create a map. It is used only when a user needs further information about any feature type on the map. However, we have found this very useful when building interactive user interfaces to geophysical applications. The GetFeatureInfo method allows us to send additional information (such as earthquake fault dimensions and material properties) to simulation codes that use these as inputs [10, 11].

The GetFeatureInfo works as follows: the user supplies an (x, y) Cartesian coordinate and the layers of interest and gets the information back in the form of HTML, GML or ASCII format. All these supported formats are defined again in WMS Capabilities file. Figure 8 illustrates the successive processes done by the WMS to respond to getFeatureInfo requests from the WMS

Client. To make the presentation more concrete in the figure, we assumed the feature information is requested in text/HTML format. This value is defined in parameter “info_format” in getFeatureInfo request. GetFeatureInfo service interface supports two more info_formats as well. These are plain text and GML formats. Since HTML creation requires a generic XSL [26] file and XSLT transformation, we have chosen this type of requests to demonstrate getFeatureInfo request processing in Figure 8. All the processes explained in Section 4.3.2 for the getMap until getting requested features from WFS are same for the getFeatureInfo processing. Again all the remote invocations are done by using SOAP messages.

After getting the feature collections data from the WFS, instead of producing map as explained in Figure 7, WMS lists all the non-geometry elements and attributes in the returned GML file. For the getMap request WMS deal with geometry elements of the returned GML file but for the getFeatureInfo WMS deal with non-geometry elements. From the list of non-geospatial elements, WMS creates a new XML file to be able to transform non-geometry elements into HTML. This XML file is simply another form of GML which includes just non-geometry elements, properties and attributes. To display all of the processes involved in getFeatureInfo handling (Figure 8), we assumed information is requested in HTML format. After creating new XML file from the non-geo elements, WMS creates HTML file from newly created XML file by using generic XSL file and XSLT transformation machine. For the detailed documentation about the getFeatureInfo, please see our project page [2].

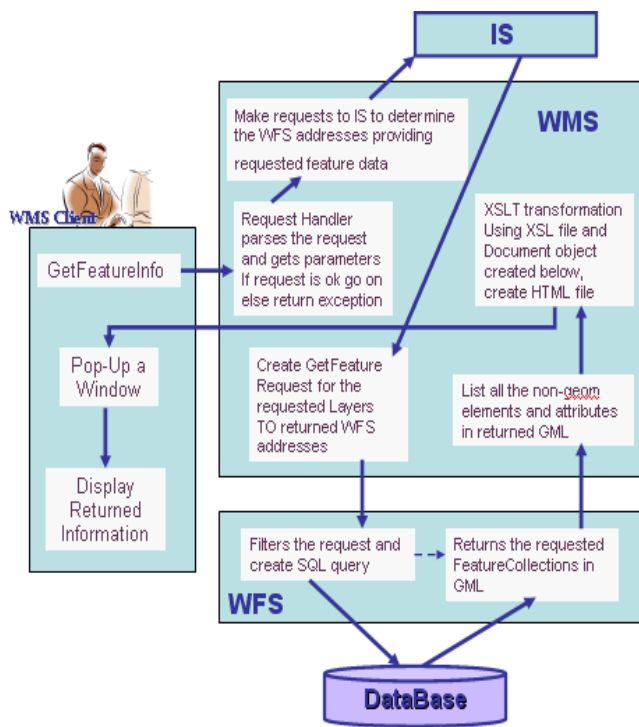


Figure 8 : getFeatureInfo work flow.

4.4 Bridging Web Service Oriented WMS to other WMS Instances

This section explains the architecture to combine Web Services based implementation of WMS systems with the third party WMS systems. Third party systems use HTTP as distributed computing platform.

Cascading WMS is the key issue to enable bridging of these two groups of visualization systems. A cascading WMS is a WMS which aggregates the contents of several individual WMS into one service that can be accessed by clients. Cascading WMS acts like a client to the other WMS and as a server to the clients [4]. The client does not need to keep track of several WMS servers; it only has to be aware of one. The client application does not need to know the ultimate source of all images.

A cascading map server reports the capabilities of the other WMS as its own and aggregates the contents and capabilities of several distinct WMS servers into one service. In most cases, the cascading map server can work on different WMS servers that cannot serve particular projections and formats themselves [5].

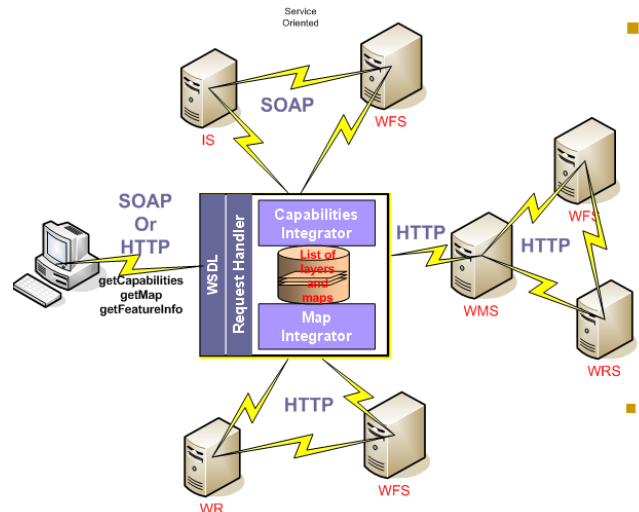


Figure 9 : Bridging of the Web Service-compatible WMS and other WMS.

Clients make their requests to cascaded WMS. Cascaded WMS services are implemented as Web Services. Clients create their requests and send them in SOAP messages over HTTP. WMS parse coming requests by request handlers. Request handlers derive all the parameters from the request and trigger the responsible modules in the WMS. Figure 9 gives a general depiction.

After getting and parsing the requests WMS defines the requested layers’ names. WMS determines if the requested layers are cascaded or not by looking at its capability file. If layer is cascaded than WMS defines the other third party WMS providing requested layer by looking at the capabilities file. If the layer is not cascaded than WMS determines the addresses of the WFS services that provide these layers by making geo-query to IS. For the cascaded layers, requests to the other (non-Web Service) WMS instances are done over HTTP as defined in OGC specifications, HTTP GET and POST.

As it is shown in Figure 9, proxy cascading WMS integrate SOAP and HTTP based GIS environments. Clients do not have to prepare different versions of requests for the different types of WMSs. Clients just send their requests to the cascading proxy WMS and get the result. In the architecture shown in Figure 9 proxy WMS can be an internal node or an external node of either HTTP based GIS networks or Web Service based GIS networks.

Figure 10 illustrates this. We have combined earthquake seismic data as feature from a WFS server with Landsat 7 satellite imagery map from WMS at NASA OnEarth [25]. WMS from OnEarth provides access to the World map via OGC compatible HTTP GET and POST requests. We are using these clients to set up geophysical simulation runs, as initially described in [10, 11].

5. GEOPHYSICAL APPLICATIONS CASE SCENARIO

In this section we first describe our WMS Client and explain the quality of services for the geophysical applications. Then we give a sample geophysical application scenario on this client.

WMS Client for the Geophysics Applications: We have developed a portlet-based browser client to our Web Service based standard visualization system for testing and the demonstration purposes. A sample WMS client is shown in Figure 10. Several capabilities are implemented for the user to access and display geospatial data. Our WMS client enables the user to zoom in, zoom out, measure distance between two points on the map for different coordinate reference systems, to get further information by making `getFeatureInfo` requests for the attributes of the features on the map, and drag and drop the map to display different bounding boxes. Users can also request maps for the area of interest by selecting predefined options clicking the drop-down list. The user interface also allows the user to change the map sizes from the drop-down lists or enable them to give specific dimensions. Zoom-in and zoom-out features let the user change the bounding box values to display the map in more or less details. Each time user change the bounding box values, user interface shows the updated bounding box values at the each side of the map.

We created generic and application independent WMS client. It can support more than one geophysics application at the same time. Each geophysics application is bound to a set of layers. These bindings are defined in structured xml properties file. Users navigate over the applications by selecting set of layers from the dropdown list. Set of layers in the dropdown list created according to communicated WMS. Binding properties are updated based on the set of supported layers of the communicated WMS.

Our implementation of the client is modular. In order to interact with a specific geophysics application we integrate a plug-in with a modular client. In order to interact with corresponding geophysics application, each component adds various application specific features to WMS client. Each plug-in can be defined by user it creates a sort of abstraction layer where users can define how to interact with geophysics application.

We created our visualization client to interact with Web Services based visualization systems (architecture is explained in Section 4) but it can also be used for the HTTP based OGC WMSs.

Client interface gives the end users lots of functionality required by the geophysics applications by using Java Server Pages (JSP), Cascading Style Sheets (CSS) and Java Script technologies. We have also developed a portlet version of the WMS Client to be able to deploy in a JSR 168-compatible portlet container. This simplifies distribution of our client application.

WMS services are stateless services. Each time a user makes a request, the WMS client creates a new request object and invokes remote WMS. All the requests are created according to schema files defined in Section 0 and wrapped into the SOAP envelope. After creating SOAP message it is sent over HTTP to the remote WMS.

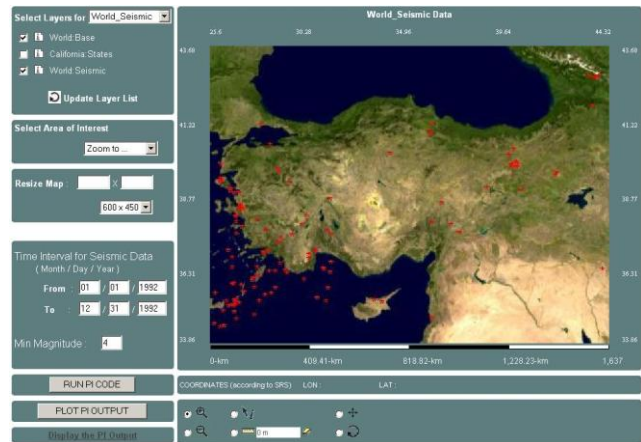


Figure 10: Project Demo page with the geophysics application. It uses Turkey’s Earthquake Seismic data.

We started to upgrade the client and visualization architecture to provide scientific visualizations, real time streaming, and collaborative mappings. For the detailed future works please see the Section 6.

Sample Geophysical Application scenario (PI): We use proposed visualization architecture for the Pattern Informatics (PI) geophysics applications [31] in the SERVOGrid project [30]. SERVOGrid project integrates historical, measured, and calculated earthquake data with simulation codes. SERVOGrid resources are located at various institutions across the country. The SERVOGrid Complexity and Computational Environment (CCE) [32] is an environment to build and integrate different domains of Grid and Web Services into a single cooperating system. As a part of SERVOGrid CCE environment, we chose the PI application which is used to produce the well-publicized “hot spot” grid-values published by SERVO team member Prof. John Rundle and his group at the University of California-Davis. Hot spot values are returned from a remote server running PI algorithms.

In this geophysics application scenario, WMS client gets the output of the PI server as grid-values, interprets it and overlay as another layer over the current map. Overlay layer for the hot spots is created by assigning different colors for each grid cell according to their values (Figure 11). These colors represent the different ranges of probabilities of the earthquake for the seismic point in the future. These jobs are done by the PI module deployed in WMS client. As we mentioned before, each

geophysics application has its own module to fulfill the application specific tasks.

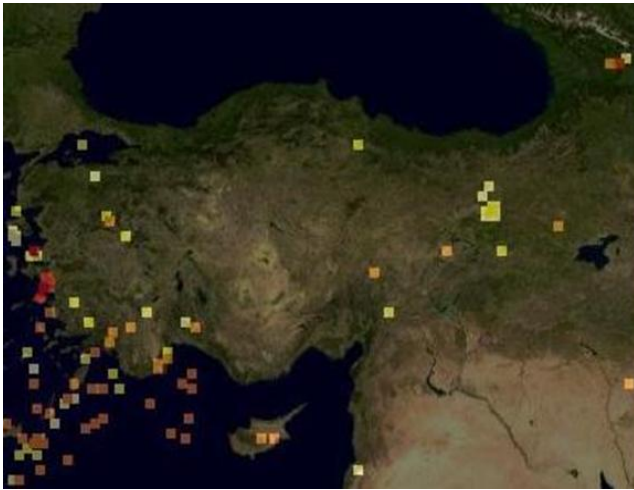


Figure 11: Overlaid layer created by PI module in WMS client after running PI geophysics application over the map displayed in Figure 10.

6. FUTURE WORK

We plan to implement Web Coverage Service (WCS) [13], Coverage Portrayal Service (CPS) [20] and Styled Layer Descriptor (SLD) Service. All these services have corresponding OGC specifications and they should be implemented according to the specifications to become OGC compatible. All should have well defined service interface described in their WSDL files. Each of these services can be implemented as a standalone application, but we will be deploying them in our project step by step. First we will finish implementation according to specifications and then handle the interoperability issues between these and already used OGC services.

We plan to use our WMS services for scientific visualization. To be able to adapt WMS to scientific visualization we need to handle high volume of data. This requires us to solve performance problems by motivating distributed High Performance Computing and collaborative shared WMS supporting multiple simultaneous Clients.

We will be working on optimization and performance algorithms of the system. To accomplish this, we will need to handle image pipelining, faster rendering, caching or client rendering.

7. CONCLUSION

The spatial data between different districts and different departments need to be shared and to be made interoperable. ISO/TC211 and OGC have defined interface specifications and standards to ensure sharing and interoperable capability of the spatial data. By adapting these to Web Service standards, we

simplify the interoperation of GIS services with other service domains.

In this document we have described our efforts to build an OGC compatible GIS Services by using Web Service technologies and OGC specifications.

The Web Services model of the GIS systems provides users with just the services and data they need, without having to install, learn, or pay for any unused functionalities. Geophysical applications such as SERVGrid project involve various kinds of data processors and data providers distributed geographically. By using service oriented GIS architecture, we can integrate new servers into our geophysics applications seamlessly. Web services are platform neutral, operating system neutral, language neutral and easily extendable.

We can extend OGC OpenGIS specifications as much as we can, but we need to consider the performance issue. This will be an important issue for us in upcoming work. Since images and capabilities documents can be too large and transferring these data over the internet is cumbersome, our first priority will be researching techniques for improving WMS performance. Visualization can be slow as overlays or even basic maps become large. Complicated maps also require large capabilities files, and parsing these can be a bottleneck. Such efficiency and performance issues will be important to our investigations of streaming map servers.

8. ACKNOWLEDGEMENTS

This work is supported by the Advanced Information Systems Technology Program of NASA's Earth-Sun System Technology Office and the National Science Foundation's National Middleware initiative.

9. REFERENCES

- [1] OGC (Open Geospatial Consortium) official web site <http://www.opengeospatial.org/>
- [2] GIS Research at Community Grids Lab, Project Web Site: <http://www.crisisgrid.org>.
- [3] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. "Web Service Architecture." W3C Working Group Note, 11 February 2004. Available from <http://www.w3c.org/TR/ws-arch>.
- [4] Jeff De La Beaujardiere, OpenGIS Consortium Web Mapping Server Implementation Specification 1.3, OGC Document #04-024, August 2002.
- [5] Kris Kolodziej, OGC OpenGIS consortium, OpenGIS Web Map Server Cookbook 1.0.1, OGC Document #03-050r1, August 2003.
- [6] Lalonde, W. (ed.), Styled Layer Descriptor(SLD) Implementation Specification 1.0.0, OGC Document #02-070, August 2002

- [7] Vretanos, P. (ed.), Web Feature Service Implementation Specification (WFS) 1.0.0, OGC Document #02-058, September 2003.
- [8] Ahmet Sayar, Marlon Pierce, Geoffrey Fox OGC Compatible Geographical Information Services Technical Report (Mar 2005), Indiana Computer Science Report TR610
- [9] Simon Cox , Paul Daisey, Ron Lake, Clemens Portele, Arliss Whiteside, Geography Language (GML) specification 3.0, Document #02-023r4., January 2003.
- [10] Galip Aydin, Marlon Pierce, Geoffrey Fox, Mehmet Aktas and Ahmet Sayar "Implementing GIS Grid Services for the International Solid Earth Research Virtual Observatory". Submitted to Journal of Pure and Applied Geophysics.
- [11] Mehmet Aktas, Galip Aydin, Andrea Donnellan, Geoffrey Fox, Robert Granat, Lisa Grant, Greg Lyzenga, Dennis McLeod, Shrideep Pallickara, Jay Parker, Marlon Pierce, John Rundle, Ahmet Sayar, and Terry Tullis "iSERVO: Implementing the International Solid Earth Research Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services" Technical Report December 2004, to be published in Special Issue for Beijing ACES Meeting July 2004.
- [12] John D. Evans, OGC Web Coverage Service (WCS) Specifications 1.0.0, Document #03-065r6 August 2003
- [13] Jérôme Sonnet, Charles Savage. OGC Web Service Soap Experiment Report 0.8 Document#03-014, Jan 2003.
- [14] Douglas Nebert, Arliss Whiteside, OpenGIS Consortium Catalogue Services Specifications 2.0. OGC Document# 04-021r2, May 2004.
- [15] Fran Berman, Geoffrey C, Fox, Anthony J. G. Hey., Grid Computing: Making the Global Infrastructure a Reality. John Wiley, 2003.
- [16] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Dave Winer., Simple Object Access Protocol (SOAP) Version 1.1, May 2000.,
- [17] Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, Web Service Description Language (WSDL) Version 1.1, March 2001.
- [18] Ferraiolo, Dean Jackson, Scalable Vector Graphics (SVG) Specification 1.1., January 2003.
- [19] Mark Little, Eric Newcomer, Greg Pavlik., OASIS Web Services Context Specifications (WS-Context) 0.8. November 2004.
- [20] Jeff Lansing., OWS1 Coverage Portrayal Service (CPS) Specifications 1.0.0, Document #02-019r1 February 2002.
- [21] A Note on Distributed Computing, S. C. Kendall, J. Waldo, A. Wollrath, G. Wyant, A Note on Distributed Computing, Sun Microsystems Technical Report TR-94-29, November 1994. Available from <http://research.sun.com/techrep/1994/abstract-29.html>.
- [22] Web Services Technologies <http://www.w3.org/2002/ws/>.
- [23] Foster, I. and Kesselman, C., (eds.) The Grid 2: Blueprint for a new Computing Infrastructure, Morgan Kaufmann (2004).
- [24] Message based middleware project at Community Grids Lab, Project Web Site: <http://www.naradabrokering.org/>
- [25] Project OnEarth at NASA JPL (Jet Propulsion Lab) <http://onearth.jpl.nasa.gov/>
- [26] W3C XSL Web Site : <http://www.w3.org/Style/XSL/>
- [27] Aydin G., SERVOGrid WFS implementation web page: <http://www.crisisgrid.org/html/wfs.html>
- [28] Aktas M., SERVOGrid Information Services Web Site, <http://grids.ucs.indiana.edu/~maktas/fthpis>
- [29] Bellwood, T., Clement, L., and von Riegen, C. (eds) (2003), UDDI Version 3.0.1: UDDI Spec Technical Committee Specification. Available from <http://uddi.org/pubs/uddi-v3.0.1-20031014.htm>
- [30] Marlon Pierce, Choonhan Yoon and Geoffrey Fox: Interacting Data Services for Distributed Earthquake Modeling. ACES Workshop at ICCS June 2003 Australia.
- [31] 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).
- [32] Geoffrey Fox, et al, Complexity Computational Environment (CCE) Architecture. Technical Report available from <http://grids.ucs.indiana.edu/ptliupages/publications/CCE%20Architecture.doc>
- [33] Alameh N., Chaining Geographic Information Web Services, IEEE Internet Computing, Sept-Oct 2003, 22-29.
- [34] Open GIS Consortium Inc. OWS-1 Registry Service. 2002-07-26.