# MESSAGE EXCHANGES FOR WEB SERVICE-BASED MAPPING SERVICES

Ahmet Sayar [1, *], Marlon Pierce [1] and Geoffrey Fox [1, 2, 3, 4]
[1] Community Grids Lab, Indiana University, Bloomington, Indiana, 47404, USA
[2]Department of Computer Science, Indiana University
[3]Department of Physics, Indiana University
[4]School of Informatics, Indiana University
{asayar, mpierce, gcf}@cs.indiana.edu

**ABSTRACT**

The Open Geospatial Consortium (OGC) [1] defines a number of standards, both for data models and for online services, that has been widely adopted in the Geographical Information System (GIS) community. This has lead to a number of software development efforts, online data archives, and application communities. The emergence of Web Service technique overcomes the shortcoming of traditional Distributed Object technique and provides the interoperable capability of cross-platform and cross-language in distributed net environment. GIS services will be implemented more extensively by using Web Service approach. A spatial data infrastructure lets many GIS vendors share data stores and applications in a distributed environment. GIS basically involves the integration of data and services from multiple sources from different vendors. The Web services architecture establishes a standard interconnection rules between services and information clients that nicely support the dynamic integration of data, which is the key to creating a spatial data infrastructure. By introducing Web Services, distributed GIS services from different vendors can be dynamically integrated into the GIS applications using the interoperable standard communication protocols of the Web Services. To be able to benefit from the Web Services in the GIS applications, all the service providers should provide their services as Web Services. General acceptance from the vendors increases the interoperability and enhances the GIS applications. We find that the OGC standards are very compatible with Web Services standards, although they are not technically implemented this way. To be able to benefit from Web Services technologies we have built a common architecture to convert any OGC GIS services to Web Services and applied this to our current WMS project.

**KEY WORDS:** Geographic Information Services (GIS), Web Map Services (WMS), Web Feature Services (WFS), Geographic Markup Language (GML), Web Services, Open Geospatial Consortium (OGC).

## 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. To solve the interoperability problems, the Open Geospatial Consortium (OGC) [1] 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.

However, we believe there is a larger interoperability problem that must be addressed. GIS services, such as defined by the OGC, are part of a larger effort to build distributed systems, such as Grids [15, 26], 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 [25]. Web Service architectures [3] are a common implementation of SOA ideals, and Grid computing has converging requirements [15, 26]. By implementing Web Service versions of GIS services, we can integrate them directly with scientific application grids [11].

This document describes our Web Service re-factoring of OGC specifications for the Web Map Service. We focus here on message exchanges and service interface design, which may be used to build clients to our map service without knowing implementation details. These implementation details are described in companion publications [8, 32]. This is part of a larger effort by our group to investigate translations of GIS services into Web Service standards [2]. In addition to our work, other Web Service compatible WMS development is documented in [13].

## 2. WEB SERVICES FOR GEOGRAPHICAL INFORMATION SYSTEMS

Web Services are part of a general, service oriented approach to distributed computing. Web Services are self-contained, self-describing, and modular. Unlike earlier, more tightly coupled distributed object approaches such as CORBA, Web Service systems support an XML message-centric approach, allowing us to build loosely coupled, highly distributed systems that span organizations. Web Services also generalize many of the desirable characteristics of GIS systems: Web Services standards provide general purpose

* Corresponding author.

specifications for publishing, locating, and invoking services across the Web. Once the service is deployed, other applications can discover and invoke the deployed service. 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) [19]. WSDL represents information about he interface and semantics of how to invoke or call a Web service. There are four important pieces of information about a Web Service. First, it contains interface information describing all the available operations (or functions). Second, it has information about the data types for incoming and outgoing messages to these operations. Third, it provides binding information about the protocols to be used for invoking the specified Web Service. Last, it contains address information for locating the specified Web Service.

Other systems interact with the Web Service in a manner as described in WSDL using Simple Object Access Protocol (SOAP) messages. SOAP [18] 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. SOAP is used in combination with transport protocols such as HTTP. Services commonly follow remote procedure call conventions, but we expect this to give way to other message exchange patterns and distributed SOAP message routing as SOAP version 1.2 implementations become available. Software messaging systems such as NaradaBrokering will enable these next-generation Web Service systems [35, 36].

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. Some of these features are being developed by using Web Service infrastructure in NaradaBrokering [37], message based middleware system, developed in 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.

We now briefly review the steps needed to develop a GIS Web Service:

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 HTTP POST and HTTP GET requests defined in 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 3.2 for the WMS case)

We outlined two different distributed computing platforms in Figure 1 and Figure 2. One is used by the OGC and defined as a standard in its specifications and the other one is used by the Web Services.

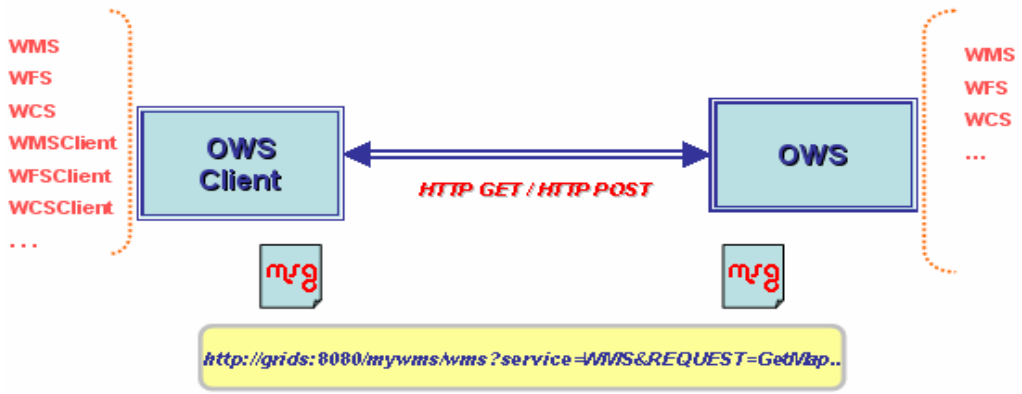


**Figure 1: Current OGC specifications use HTTP GET/POST to relay requests to services.**

Figure 1 outlines the distributed computing platform for the message exchange between OGC GIS clients and servers. The Online Resource of each operation supported by an OGC server is an HTTP Uniform Resource Locator (URL). The URL may be different for each operation, or the same, at the discretion of the service provider. Each URL conforms to the description in IETF RFC 2616. Only the query portion comprising the service request itself is defined by the OGC WMS specification [4]. HTTP supports two request methods: GET and POST. One or both of these methods may be offered by a server, and the use of the Online Resource URL differs in each case. Support for the GET method is mandatory; support for the POST method is optional.

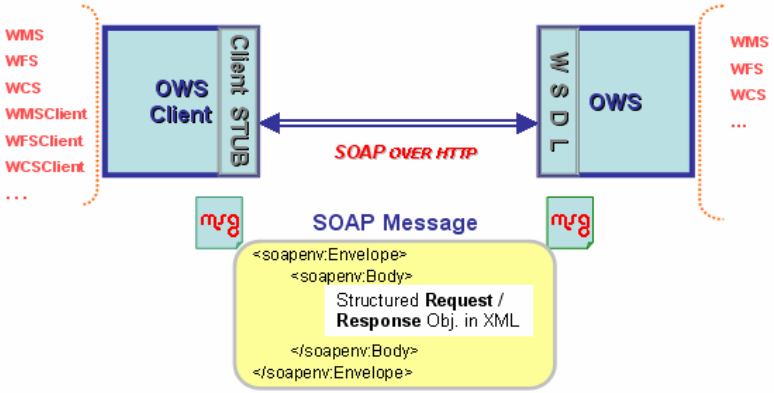


**Figure 2: Web Service invocations communicate with structured message in extensible SOAP envelopes.**

When the server is implemented as Web Service then the message exchange will be done by the SOAP over HTML. Web Services are invoked by the standardized XML messaging system. SOAP is an XML-based protocol for exchanging information between servers. Although SOAP can be used in a variety of messaging systems, and can be delivered via a variety of transport protocols, the main focus of SOAP is RPCs transported via HTTP. Sample requests replaced with the white box of the SOAP message in Figure 2 is given in Figure 4, Figure 5 and Figure 6. These are schema files to invoke WMS functionalities implemented as Web Services. Request instances are created according to these schema files. Name, number and type of the request parameters are checked for the validity and well-formedness in the server side. SOAP carries these structured requests in its body part.

Servers running Web Services publish their public interfaces defined in WSDL files. WSDL is an XML grammar for specifying public interface for a Web Service. This public interface can include information on all publicly available functions, data type information for all XML messages, binding information about the specific transport protocol to be used, and address information for locating the specified service.

OGC has experimental report and discussion papers [13, 20] on some mature GIS services about the Web Services and SOAP but there is no standard implementation specifications and interface definitions for all of the OWSs. Here we outlined the general ways to be able to make an OGC compatible GIS server a Web Service. In the next section, we outline the message exchange formats and the service interfaces for using the Web Map Service. This is taken from the real WMS implementation done for the ServoGrid application [33].

## 3. SERVICE INTERACTIONS AND MESSAGING ARCHITECTURE OF WMS WEB SERVICES

The WMS OpenGIS Specification specifies the implementation and use of the WMS operations (GetCapabilities, GetMap and GetFeatureInfo) in the Hypertext Transfer Protocol (HTTP) distributed computing platform. WMS operations can be invoked using a standard web browser by submitting requests in the form of Uniform Resource Locators (URLs). In the specification it is also said that future version may apply to other distributed computing platforms such as Web Services.

As discussed above, Web Services use SOAP for messaging. SOAP is an XML protocol. SOAP provides an envelope that encapsulates XML data for transfer through the web infrastructure (e.g. over HTTP, through caches and proxies). Most services (including our own) use the Remote Procedural Call (RPCs) encoding convention, but we expect more message-centric applications in the future with the release of Apache Axis 2 and other implementations of SOAP Version 1.2. Serialization mechanisms are based on XML Schema data types.

### 3.1 Mapping OGC WMS to WSDL

The Web Map Service produces maps from geographic data. Maps create information from raw geographic data, usually obtained from related GIS services such as the Web Feature Service. Maps are generally rendered in pictorial formats such as JPEG, GIF, and PNG. WMS also produce maps from vector-based graphical elements in Scalable Vector Graphics (SVG).

There are two types of WMS defined in the specifications. These are basic WMS and SLD-enabled WMS. For the basic WMS, there are three operations defined. These are getCapabilities, getMap, and GetFeatureInfo. If the WMS is SLD-enabled then there will be four more operations supported, describeLayer, getLegendGraphics, getStyles and putStyles. DescribeLayer is used for asking an XML description of a map layer. GetLegendGraphics is used for acquiring the legend symbols. GetStyles is used for retrieving user-defined styles from WMS. PutStyles is used for storing user-defined styles into WMS. In this document, from now on, we will be just mentioning about the basic WMS.

HTTP is the distributed computing transport protocol supported by the OGC WMS specification. HTTP supports two request methods, GET and POST. One or both can be supported by the WMS and at each case URL format changes. Support for the GET method is mandatory but support for the POST method is optional. WMS operations are invoked by submitting requests in the form of Uniform Resource Locators (URLs). The content of these URLs depends on the operations and the parameters of the requests.

WMS publishes its ability and data holdings in its capabilities document. This document is encoded in XML. WMS classifies its geographic data holdings in the "Layers" and gives information about the styles available for these Layers. Each layer can have sub layers and the sub layers can have different styling defined for them. [4].

In our Web Service-based implementation of the Map Server, we wanted to take advantage of Web Service's ability to transmit structured XML messages rather than rely solely on HTTP Request name/value pairs. These formatted messages will allow us to check the structural validity of the message using standard XML parsers and to bind the messages to (Java) data objects using data binding frameworks for simpler manipulation. Both the parsing tools and the data binding frameworks represent general purpose XML tools that can be leveraged in Web Service approaches. We note also (in future work), Web Services are message-centric and may be implemented using message-oriented middleware, so we anticipate this development in our approach.

In Table 1 and Table 2, we summarize our mappings of OGC WMS messages (requests and responses) to WSDL interface of Web Service based WMS. Mappings in these tables are applied to current WMS implementation [29].

| **WSDL MESSAGES IN FIGURE 3** | **OGC REQUEST DEFINED IN WMS SPEC[4]** | **REQUESTS TO WMS WEB SERVICES** |
|---|---|---|
| getCapabilityRequest | HTTPGET/POST SERVICE=WMS REQUEST=getCapabilities | Instance of schema file in Figure 4 wrapped into SOAP |
| getMapRequest | HTTPGET/POST SERVICE=WMS REQUEST=getMap | Instance of schema file in Figure 6 attached to SOAP |
| getFeatureInfoRequest | HTTPGET/POST SERVICE=WMS REQUEST=getFeatureInfo | Instance of schema file in Figure 5 attached to SOAP |

**Table 1: Mapping OGC-WMS Requests to WMS Web Services defined in WSDL**

The WSDL interface of our WMS is shown in Figure 3. For brevity, we only show the sections (messages and portTypes) relevant to our current discussion. The first column of Table 1 lists "request" messages supported by the service based WMS. The second column lists the corresponding OGC compatible WMS requests and related parameters to define the request types. The last column lists OGC compatible requests to Service based WMS to invoke the services described in Figure 3.

| **WSDL MESSAGES IN FIGURE 3** | **OGC WMS RESPONSE TYPE** | **RETURN TYPES SUPPORED BY OUR WMS** | **RESPONSES OF WMS SERVICES IN WSDL FIGURE 3** |
|---|---|---|---|
| getCapabilityResponse | Defined in request parameter<br>Name : 'FORMAT' :<br>Value: (MIME type) default text/xml | - text/xml | String –<br>data type on the left column is written into String |
| getMapResponse | Defined in request parameter<br>Name : 'FORMAT' :<br>- (MIME type) no default | - image/svg<br>- image/jpeg | Object –<br>data types on the left column are written into datahandler obj |
| getFeatureInfoResponse | Defined in request parameter<br>Name : 'INFO_FORMAT'<br>- (MIME type) no default | - text/plain<br>- text/HTML<br>- application/vnd.ogc.gml | String –<br>data types on the left column are written into String |

**Table 2: Mapping OGC-WMS Responses to WMS Web Services Response Messages defined in WSDL**

Table 2 defines the mappings of OGC WMS responses to Service based WMS responses. First column of Table 2 lists response messages supported by the service based WMS (see Figure 3). Second column lists corresponding OGC compatible WMS request messages and related parameters to define the response types. Currently we have Service based WMS and it has a Capabilities file. In its Capabilities file we have defined supported response types for each of the requests and listed them in third column. Last column lists the actual response types to corresponding requests and supported response types in terms of OGC specifications. Return types defined as Strings are structured data in XML Strings. Strings are actually xml, plain text, HTML or GML depending on the requested format.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://toro.ucs.indiana.edu:8092/wms/services/WMSServices"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://toro.ucs.indiana.edu:8092/wms/services/WMSServices"
xmlns:intf="http://toro.ucs.indiana.edu:8092/wms/services/WMSServices"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://lang.java" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types><schema targetNamespace="http://lang.java" xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
<complexType name="Object"><sequence/></complexType></schema></wsdl:types>
  <wsdl:message name="getMapResponse">
    <wsdl:part name="getMapReturn" type="tns1:Object"/>
  </wsdl:message>
  <wsdl:message name="getFeatureInfoResponse">
    <wsdl:part name="getFeatureInfoReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getCapabilityRequest">
    <wsdl:part name="request" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getMapRequest">
    <wsdl:part name="request" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getFeatureInfoRequest">
    <wsdl:part name="request" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getCapabilityResponse">
    <wsdl:part name="getCapabilityReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="WMSServices">
    <wsdl:operation name="getMap" parameterOrder="request">
      <wsdl:input message="impl:getMapRequest" name="getMapRequest"/>
      <wsdl:output message="impl:getMapResponse" name="getMapResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getCapability" parameterOrder="request">
      <wsdl:input message="impl:getCapabilityRequest" name="getCapabilityRequest"/>
      <wsdl:output message="impl:getCapabilityResponse" name="getCapabilityResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getFeatureInfo" parameterOrder="request">
      <wsdl:input message="impl:getFeatureInfoRequest" name="getFeatureInfoRequest"/>
      <wsdl:output message="impl:getFeatureInfoResponse" name="getFeatureInfoResponse"/>
    </wsdl:operation>
  </wsdl:portType>
 +<wsdl:binding name="WMSServicesSoapBinding" type="impl:WMSServices">
 +<wsdl:service name="WMSServicesService">
</wsdl:definitions>
```

**Figure 3: Primary sections of WMS WSDL service interface description. (*Available at http://toro.ucs.indiana.edu:8092/wms/services/WMSServices?wsdl*)**

Future WMS services will address issues with return types. WMS service interfaces will be evolving with detailed return types. Response types listed in WMS Services are actually structured complex data types encoded in XML. For example for the getFeatureInfo request, after creating feature information in text/HTML format, WMS puts it into SOAP message as payload in the form of String. Nothing is changed in the structure of the response in HTML.

### 3.2 Client Interaction to Web Service Based WMS

WMS Clients are not only browser-based, but also other WMS or command line applications. In other words WMS functions can be invoked over the Internet from the command line or from the browser but all the requests should be well defined and well formed. Validation and well formedness checks are done by the request parsers at the WMS server side. This is an advantage of encoding requests in XML. XML enables encoding of the complex data types.

All the valid request instances to WMS should be created in accordance with the request schemas defined in below figures. Since the entire schema files are created according to the rules and restrictions in WMS Specifications, all the requests created according to

these schema files can be used to invoke Web Services based OGC WMS services. Requests have some parameters whose names, numbers, and values assigned to them should obey the rules defined in the specifications [4] to be OGC compatible. We can thus (after processing the SOAP message), extract and recreate URL-based invocations for backward compatibility with existing Web Map Servers. We use this to develop support for cascading WMS implementations, described below.

Below schema files are created with the help of Altova XmlSpy.

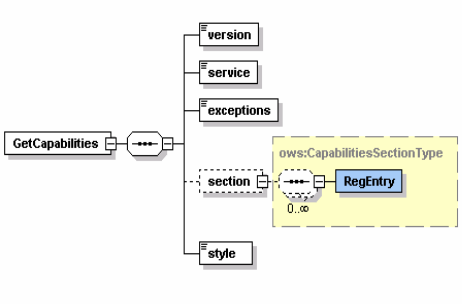


**Figure 4 : GetCapabilities Request Schema.**

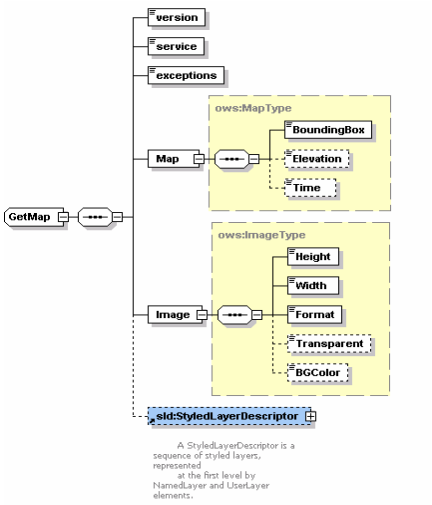


**Figure 6 : GetMap Request Schema.**

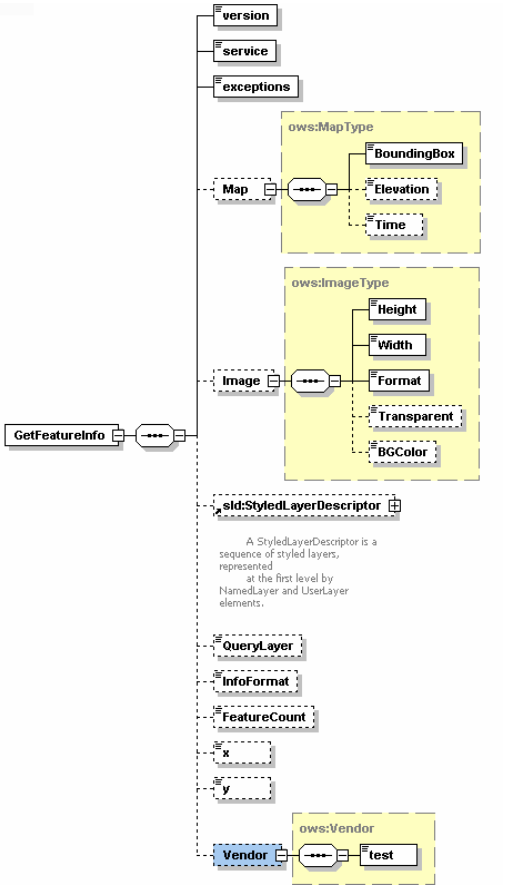


**Figure 5: GetFeatureInfo Request Schema.**

Clients create these requests after getting required parameter from the user. Web Services have their own client stubs which are created from their public service interfaces. All the clients to any specific service should first create client stubs to be able to invoke this service. Many web service implementations provide tooling to simplify this step. For example, Apache Axis's WSDL2Java enable clients to create client stubs from the service interface defined in WSDL files.

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 wrapped into the SOAP envelope. After creating SOAP message it is sent over HTTP to the remote WMS. Invocation is done by the WMS Client-stubs created by using Apache Axis 1.2 [34].

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. For the message descriptions in WMS service interface, see the Figure 3.

### 3.3 Bridging Web Service Oriented WMS to other WMS

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 GIS 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 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].

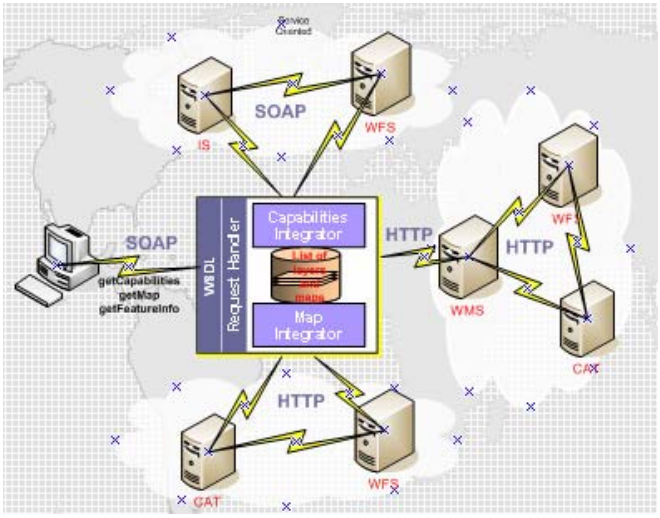


**Figure 7 : Bridging of the Web Service-compatible WMS and other WMS.**

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 7 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.

Figure 8 illustrates this. We have combined earthquake seismic and state-boundaries data as features from a WFS server with Landsat 7 satellite imagery map from WMS at NASA OnEarth [27]. 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 [11, 32]

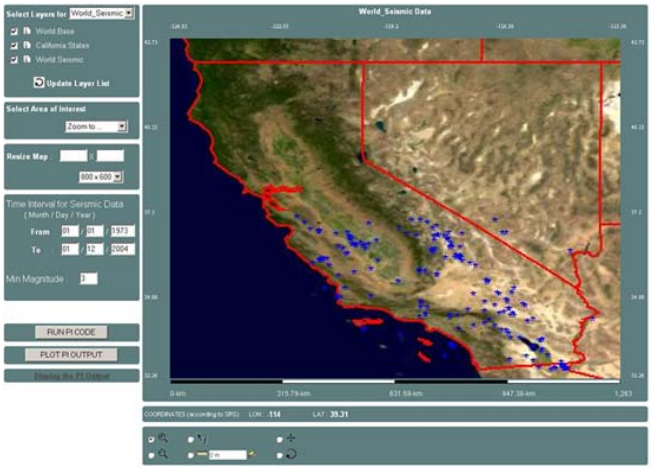


**Figure 8: WMS Client with the geophysics application. Displayed layers are California Seismic and State-boundary layers.**

## 4. FUTURE WORK

We have explained the conversion of the OGC compatible GIS services to Web Services. We have implemented Web Service versions of the OGC's Web Map Service and Web Feature Service specifications. We plan to implement other GIS services, such as Web Coverage Service (WCS) [13], Coverage Portrayal Service (CPS) [24] and Styled Layer Descriptor (SLD) Service, to enhance our project All these services have corresponding OGC specifications and can be adapted using our established approach.

In addition, we plan to build new interconnection infrastructure for the message delivery between these GIS services, to eliminate the service dependence on HTTP for SOAP transport. We use NaradaBrokering as a message based middleware system between these components to deliver the request and response messages wrapped in SOAP envelope. NaradaBrokering is a distributed messaging infrastructure that provides a message oriented middleware which facilitates communications between the distributed entities through the exchange of messages. NaradaBrokering provides some features that are important in GIS area. These are Quality of Service (QoS) and security profiles for sent and received messages, interface with reliable storage for persistent events, reliable delivery via WS-Reliable messaging, fault tolerant data transport, support for different underlying transport implementations such as TCP, UDP, Multicast, SSL, RTP, HTTP, discovery service to find nearest brokers / resources (efficient routing).

Future work also includes transcoding of the map images into video streams for portraying time-series data. Map images will be dynamically generated from the time series geographic data. On the fly, those images will be transcoded into video streams, which can be in H.261, H.263 or MPEG formats. Video streams could be received and played by clients in GlobalMMCS [38, 39] or AccessGrid [40] sessions. We plan to create collaborative video mappings for the scientific geophysics applications grids.

While we are trying to achieve these near and long term targets, we will always keep the performance issues in mind. Since GIS applications involve huge amount of spatial data transfers, the performance is always hot topic in GIS area. To improve the performance we need to handle common problems in the GIS. We are planning to make a contribution to solution by generating new algorithms, generating new optimization techniques, using distributed rendering and tiling, parallel rendering of images, etc. 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.

## 5. CONCLUSION

The emergence of Web Service technique overcomes the shortcoming of other approaches. Unlike URL-based approaches uses in earlier OGC specifications, Web Services can encode and exchange structured (XML) messages that can ultimately be decoupled from underlying transports such as HTTP. Likewise, Web Services avoid problems of more tightly coupled distributed object techniques and provides the interoperable capability of cross-platform and cross-language in distributed, Internet scalable environment. Web services allow the exposure of an application programming interface over the web to be accessed remotely. Web services are implemented using Internet standards (XML, SOAP, and WSDL), thus holding the promise of wider use than competing technologies. By combining relatively simple individual Web Services with other distributed services, a more complex service can be provided in a platform independent way. GIS services implemented as Web Services can interoperate with another service based GIS services.

In this paper, we have explained our efforts to build OGC-based GIS Services by using Web Service technologies and OGC specifications. We saw that making OGC services Web Services compatible is not easy task. We have tried to picture out a generic conversion schema but it is not totally applicable in the real world. OGC has some vendor specific parameters, each vendor can describe its own service interface as WSDL file but when users tried to create Client stubs every user gets different stubs if WSDL files from different vendors are not exactly same. In addition to this, we are not allowed to change the formats of the OGC Capabilities documents not to disrupt the OGC compatibility properties of a server. We cannot add any service address whose service interface is defined as WSDL. So in order to be able to solve this kind of problems, we have used "cascaded" structure and bridging services in our proposed architecture in Web Service compatible GIS services.

In addition, as we have done in bridging service, we can extend OpenGIS specifications as much as we can but we need to consider the performance and compatibility issues. This is especially important for combining GIS services with Grid-based scientific applications, which generate and consume large amounts of data. For the Web Map Server, images of large data sets can be too large, capabilities documents can be too large and transferring these data over the internet is a cumbersome. Our first priority will be researching techniques for improving the performances of the GIS services in the project.

## 6. 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.

## 7. 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. Available at http://www.opengeospatial.org/docs/03-014.pdf

[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] Castor http://castor.exolab.org

[17] XMLBeans (http://xml.apache.org/xmlbeans )

[18] Don Box, David Ehnebuske, Gobal Kakivaya, Andrew Layman, Dave Winer., Simple Object Access Protocol (SOAP) Version 1.1, May 2000,.

[19] Christiensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, Web Service Description Language (WSDL) Version 1.1, March 2001.

[20] Philippe Duschene, Jerome Sonnet, WMS Change Request: Support for WSDL and SOAP, OGC Document #04-050r1, April 2005. Available at http://portal.opengeospatial.org/files/index.php?artifact_id=9541

[21] George Percivall, OpenGIS Consortium Reference Model 0.1.3, OGC Document #04-040, September 2003.

[22] Roel Nicolai, The OpenGIS® Abstract Specification, Topic 2: Spatial referencing by coordinates 2.0.0. Document #03-073r1, October 2003.

[23] SERVOGrid IS implementation. Available at http://grids.ucs.indiana.edu/~maktas/fthpis/index.html

[24] Jeff Lansing., OWS1 Covarage Portrayal Service (CPS) Specifications 1.0.0, Document #02-019r1 February 2002.

[25] 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.htmlKkk

[26] Foster, I. and Kesselman, C., (eds.) The Grid 2: Blueprint for a new Computing Infrastructure, Morgan Kaufmann (2004)

[27] Project OnEarth at NASA JPL (Jet Propulsion Lab) http://onearth.jpl.nasa.gov/

[28] Panagiotis A. Vretanos, OpenGIS Filter Encoding Implementation Specification 1.0.0. OGC Document #02-059, September 2001. Available at http://www.opengeospatial.org/specs/?page=specs

[29] SERVOGrid WMS implementation. Available at http://complexity.ucs.indiana.edu/~asayar/wms

[30] SERVOGrid WFS implementation. Available at http://www.crisisgrid.org/html/wfs.html

[31] Galip Aydin, Marlon Pierce, Geoffrey Fox, "High Performance Web Feature Service Implementation for GIS Grid and Web Service Architectures", Submitted to GML And Geo-Spatial Web Services Conference 2005.

[32] Ahmet Sayar, Mehmet S. Aktas, Galip Aydin, Geoffrey Fox and Marlon Pierce, “Developing a Web Service-Compatible Map Server for Geophysical Applications”, Submitted to ACM-GIS Conference 2005.

[33] 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 of Pure and Applied Geophysics ( PAGEOPH ) for Beijing ACES Meeting July 2004.

[34] Apache Axis Project Web Site : http://ws.apache.org/axis/.

[35] Geoffrey Fox, Shrideep Pallickara and Savas Parastatidis Towards Flexible Messaging for SOAP Based Services Proceedings of the IEEE/ACM Supercomputing Conference November 2004. Pittsburgh, PA.

[36] Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, Building Messaging Substrates for Web and Grid Applications to be published in special Issue on *Scientific Applications of Grid Computing* in Philosophical Transactions of the Royal Society of London 2005.

[37] Message based middleware project at Community Grids Lab, Project Web Site: http://www.naradabrokering.org/

[38] Global Multimedia Collaboration System. http://www.globalmmcs.org

[39] Wenjun Wu, Geoffrey Fox, Hasan Bulut, Ahmet Uyar, Harun Altay “Design and Implementation of A Collaboration Web-services system”, Journal of Neural, Parallel & Scientific Computations (NPSC), Volume 12, 2004.

[40] The Access Grid Project. http://www.accessgrid.org